Trabajo Final de Grado

Grado en ingeniería de Sistemas de Telecomunicaciones

Diseño Conceptual de un Shield Multi protocolo wireless para Raspberry Pi

Alberto Jiménez Gómez

Director: Raúl González Ortiz

Escuela de Ingeniería

Universidad Autónoma de Barcelona (UAB)

Julio 2018

El tribunal de evaluación de este Trabajo Final de Grado, reunido el día _____, ha acordado conceder la siguiente calificación:

Presidente: _____

Vocal: _____

Secretario: _____

El abajo firmando, ***Raúl Aragonés Ortíz***,

Profesor de la Escuela de Ingeniería de la Universidad Autónoma de Barcelona,

CERTIFICA:
Que el trabajo al que corresponde la presente memoria ha sido realizado bajo su dirección por ***Alberto Jiménez Gómez***

Y para que conste firma la presente.

ARAGONES ORTIZ RAUL - 34742227Z

Firmado digitalmente por ARAGONES ORTIZ RAUL - 34742227Z
Fecha: 2018.06.28 23:18:08 +02'00'

Resumen del Proyecto

Título del Proyecto

Proyecto que consiste en el diseño de un Shield para Raspberry PI, capaz de trabajar con distintos protocolos Wireless. En este caso, nuestra Shield se diseñará para incorporar varios chips que funcionarán con los siguientes protocolos: LoRa, Nordic, NRF ZigBee, y adicionalmente Bluetooth y WiFi a través de las UARTS integradas en la placa de la Raspberry PI

Autor:

Alberto Jiménez Gómez

Fecha:

Julio 2018

Tutor:

Raúl Aragonés Ortíz

Titulación:

Grado en ingeniería de Sistemas de Telecomunicaciones

Resumen del Proyecto

En el proyecto que a continuación se presenta, se pretende diseñar e implementar una Shield para Raspberry PI. Para ello, en primera instancia realizaremos un estudio de los múltiples protocolos wireless existentes en el área del internet de las cosas para poder entender su funcionamiento y sus características principales, y posteriormente elegiremos los protocolos con los que vamos a trabajar. Seguidamente, haremos un estudio de mercado para ver las opciones que nos ofrece este en referencia a los chips que trabajan con dichos protocolos, y a partir de éstos, diseñaremos una PCB que permita poder ensamblar estos chips en ella, y en líneas futuras, a través de su programación, poder trabajar con ella.

ÍNDICE DE FIGURAS 8

ÍNDICE DE TABLAS 9

1. INTRODUCCIÓN 10
   1.1. Descripción 10
   1.2. Objetivos 11
   1.3. Motivación Personal 12
2. REVISIÓN DEL ESTADO DEL ARTE 13
   2.1. Protocolos de comunicación 13
      2.1.1. LoRa 13
      2.1.2. SigFox 14
      2.1.3. NB-IoT 15
      2.1.4. Bluetooth 16
      2.1.5. WiFi HaLow 16
      2.1.6. ZIgBee 17
      2.1.7. LTE-M 18
      2.1.8. Nordic Semiconductor 18
   2.2. Estudio de mercado 19
      2.2.1. Protocolo LoRa 19
      2.2.2. Protocolo SigFox 19
      2.2.3. Protocolo NB-IoT 20
      2.2.4. Protocolo Bluetooth 20
      2.2.5. Protocolo WiFi 20
      2.2.6. Protocolo LTE-M 21
      2.2.7. Protocolo Nordic Semiconductor 21
      2.2.8. Protocolo ZigBee 21
   2.3. Elección Protocolos 22
      2.3.1. Elección de los chips 23
   2.4. Conclusiones sobre el estado del arte 23
3. Planificación del proyecto 24
   3.1. Fase 1: Estudio del escenario 24
   3.2. Fase 2: Estudio de los componentes para el diseño del circuito impreso 24
   3.3. Fase 3: Diseño del circuito impreso 25
   3.4. Fase 4: Pruebas / Test 25
   3.5. Fase 5: Redacción – Presentación proyecto 25
4. Introducción a la herramienta de diseño de PCB KICAD 26
   4.1. Determinación de los componentes 26
      4.1.1. Raspberry PI model B + ... 26
         4.1.1.1. Estándar SPI 27
         4.1.1.2. Estándar UART 28
      4.1.2. Multiplexor 29
      4.1.3. Demultiplexor 30
      4.1.4. 74LVC1G3157GV 30
      4.1.5. Convertidor de nivel lógico bidireccional 31
      4.1.6. Descripción módulo RN 2483 32
      4.1.7. Conector antena SMA 34
      4.1.8. Descripción módulo XBP24BZ 34
      4.1.9. Descripción módulo NFRL2401 35
   4.2. Diseño PCB con KICAD 36
      4.2.1. Diseño de los componentes en KICAD 36
      4.2.2. Diseño del esquemático 39
      4.2.3. Creación de las huellas 41
         4.2.3.1. ¿Qué es un pad? 41
         4.2.3.2. ¿Qué es una vía? 42
      4.2.4. Editor de huellas 42
      4.2.5. PCBNew – Editor de placas de circuito impreso 46
         4.2.5.1. Reglas de diseño 46
         4.2.5.2. PCB – Consideraciones hasta llegar al diseño definitivo 48

4.2.5.3. Creación de un plano de tierra 50
4.2.6. Fabricación placa 51
4.2.7. Soldadura de los componentes de la placa 53
4.2.8. Test de comprobación 54
4.2.8.1. Test de comprobación módulo XBP24BZ 54
4.2.8.2. Comunicación con módulo NRF24L01 59
4.2.8.3. Comunicación con módulo RN 2483 60
5. Líneas futuras 62
5.1. Lenguajes de programación 62
5.1.1. Phyton 62
5.1.2. C++ 63
5.2. Librerías 63
5.2.1. Librería ArduPI 63
5.2.2. Librería PynRF24 63
5.2.3. Librería para Xbee en Phyton 64
5.2.4. Librería LoRa Gateway 64
6. Conclusiones del proyecto 65
7. Glosario 66
8. Referencias bibliográficas 68

INDICE DE FIGURAS

Figura 1: Ejemplo red inalámbrica 13
Figura 2: Fase 1 planificación 24
Figura 3: Fase 2 planificación 24
Figura 4: Fase 3 planificación 25
Figura 5: Fase 4 planificación 25
Figura 6: Fase 5 planificación 25
Figura 7: Pines GPIO de la Raspberry PI 26
Figura 8: Conexión estándar SPI 28
Figura 9: Conexión estándar UART 29
Figura 10: Multiplexor 29
Figura 11: Demultiplexor 30
Figura 12: Configuración pines 74VC1G3157GV 31
Figura 13: Diagrama lógico 74VC1G3157GV 31
Figura 14: Esquema convertidor lógico bidireccional 32
Figura 15: Canales convertidor lógico bidireccional 32
Figura 16: Diagrama de pines chip RN2483 33
Figura 17: Diagrama de bloques chip RN2483 33
Figura 18: Conector de antena SMA 34
Figura 19: Diagrama de pines chip XBP24B 35
Figura 20: Conexiones modulo NRF24L01 35
Figura 21: Diseño módulo RN2483 36
Figura 22: Diseño módulo XBP24B 37
Figura 23: Diseño convertidor de nivel lógico 37
Figura 24: Diseño multiplexor / demultiplexor 37
Figura 25: Diseño cabezal Raspberry Pi 38
Figura 26: Diseño NRF2401 38
Figura 27: Diseño conector SMA 38
Figura 28: Librerías proyecto 39
Figura 29: Esquema PCB 40
Figura 30: Control de reglas eléctricas 41
Figura 31: Huella cabezal Raspberry Pi 42
Figura 32: Huella cabezal NRF24L01 43
Figura 33: Huella 74LVC1G3157GV 43
Figura 34: Huella RN 2483 43
Figura 35: Huella XBP24B 44
Figura 36: Huella convertidor 44
Figura 37: Huella conector SMA 44
Figura 38: Asignación de huellas 45
Figura 39: Asignación referencia a cada componente 45
Figura 40: Generación Netlist 46
Figura 41: Relación Ancho de pista – Corriente máxima 47
Figura 42: Reglas de diseño esquemático 47
Figura 43: Lector de Netlist 48
Figura 44: Raspberry Pi 3 model B+ 49
Figura 45: Diseño final circuito impreso 50
Figura 46: Plano de tierra capa superior 51
Figura 47: Plano de tierra capa opuesta 51
Figura 48: Archivos Gerber generador 52
Figura 49: Plantilla para realizar el pedido de una placa de circuito impreso 53
Figura 50: Resultado final después de soldar los componentes 54
Figura 51: Red mesh en protocol ZigBee 55
Figura 52: Configuración cómo dispositivo final 56
Figura 53: Configuración cómo coordinador 56
Figura 54: Configuración de los módulos Xbee 57
Figura 55: Escenario Raspberry Pi + Shield – Arduino + Arduino shield 57
Figura 56: Mensaje enviado desde la Raspberry Pi 58
Figura 57: Recepción mensaje enviado a través de la Raspberry Pi 58
Figura 58: Diagrama de conexión entre modulo NRF24L01y Arduino 60

ÍNDICE DE TABLAS

Tabla 1: Características de los protocolos LoRa, SigFox y NB-IoT 15

Tabla 2: Características de los protocolos ZigBee, Bluetooth y Wifi 17

Tabla 3: Chips protocolo LoRa 19

Tabla 4: Chips protocolo SigFox 19

Tabla 5: Chips protocolo NB-IoT 20

Tabla 6: Chips protocolo Bluetooth 20

Tabla 7: Chips protocolo WiFi HaLow 20

Tabla 8: Chips protocolo LTE-M 21

Tabla 9: Chips protocolo Nordic Semiconductor 21

Tabla 10: Chips protocolo ZigBee 21

Tabla 11: Características conexiones inalámbricas Raspberry Pi 3 B+ 27

1. Introducción

1.1 Descripción

El proyecto en el que vamos a trabajar tiene su origen en el reaprovechamiento de la energía. A partir del calor de un dispositivo, existen ciertos dispositivos que permiten transformar este calor en energía (Energy Harvesting). Y a partir de esta energía, podemos alimentar una serie de sensores o medidores que nos permitan obtener información. Cada cierto tiempo, la información recopilada por estos dispositivos debe ser enviada a una computadora, que se encargue de su recepción, y de esta manera, poder tratar estos datos. Y es aquí donde empieza nuestro proyecto. La idea es diseñar una shield adaptable a una computadora, que en este caso será una Raspberry Pi, que permitirá en aplicaciones posteriores y futuras, obtener diferentes funcionalidades, dependiendo de cada chip, y de lo que se requiera. Para ello estudiaremos los diferentes protocolos de comunicación Wireless de los que disponemos hoy en día, y a partir de aquí, llegar a diseñar un circuito que nos permita cumplir con lo que se requiere.

Deberemos diseñar pues, un circuito impreso que nos permita conectar los diferentes circuitos integrados de comunicación con la Raspberry PI, y a través de la cual podamos interactuar con ellos.

1.2 Objetivos

1. Realizar un estudio de los diferentes protocolos de comunicación Wireless más relevantes en el mundo del IoT.

2. Realizar un estudio de mercado de los diferentes chips que este ofrece actualmente, y que posteriormente utilizaremos para el diseño de nuestra shield.

3. Seleccionar los protocolos que vamos a utilizar, y por ende, los chips que vamos a utilizar.

4. Estudiar la topología de red de los protocolos seleccionados, y como se comunican

5. Estudio de las librerías necesarias para su posterior programación en eventos futuros.

6. Elegir una herramienta de diseño de circuitos impresos para poder realizar una PCB (shield) en la que colocar nuestros chips.

7. Una vez tengamos el circuito impreso, soldar los componentes necesarios para poder conectarla a la Raspberry Pi.

8. Comprobar la correcta conexión de nuestro Shield con la Raspberry Pi.

1.3 Motivación Personal

A nivel personal, me encuentro ante lo que considero un reto. He estudiado Ingeniería de telecomunicaciones en sistema de telecomunicaciones. Estoy en mi último año de grado, y pienso que se trata de un proyecto que tiene mucho potencial, y puede sacar lo mejor de mí.

Se trata de un proyecto cuya parte más importante está orientada al diseño de sistemas electrónicos. Después de tantos años estudiando, creo que es un proyecto que puede poner a prueba mi capacidad de aprendizaje sobre un temario del que tampoco he trabajado demasiado en ello durante los estudios.

Además, he tenido la suerte de tener a Raúl Aragonés como profesor en una de las asignaturas del grado, y el grado de aprendizaje fue muy elevado, y me gustó mucho su manera de impartir las clases prácticas. Así que, también es un añadido saber que puedo desarrollar el proyecto con un profesor que aportó cosas muy positivas en mí.

Es por todo lo aquí mencionado, que decido aventurarme en este proyecto, con la ilusión de que todo salga bien, y de aprender lo máximo posible.

2. Revisión del estado del arte

En este capítulo vamos a realizar una identificación y clasificación de los protocolos de comunicación existente en el mundo del Internet de las cosas.

En 1997 fue aprobado el estándar 802.11 [1], un estándar que regulaba la transmisión de datos de forma inalámbrica para interconexión entre ordenadores y el primero de una larga serie que se definía según se vieron las enormes posibilidades de los mismos. Gracias a todos estos estándares, pero sobre todo a raíz del éxito del 802.11b [2] y su hermano mayor 802.11g [3], ahora es posible la conexión a internet por parte de innumerables dispositivos, comenzando con los ordenadores personales y continuando con otros más pequeños como los ordenadores portátiles y PDAs, o dispositivos móviles.

Figura 1: Ejemplo red inalámbrica

Es por ello, que vamos a hacer un repaso de todos los protocolos de comunicación que permiten el uso de estas tecnologías Wireless, haciendo especial hincapié en aquellas que cumplan con los requisitos que vamos a necesitar. Necesitaremos que sean protocolos de bajo coste, en primer lugar, y en segundo lugar, y no menos importante, que sean protocolos cuyos chips funcionen con un consumo muy reducido.

A continuación se van a mostrar diferentes protocolos de comunicación inalámbricos, y hablaremos de sus características principales, y de las ventajas y desventajas de su uso.

2.1 Protocolos de comunicación

2.1.1 LoRa

LoRa es un protocolo de comunicación radio de baja potencia, muy utilizado en el mundo del IoT [4]. EL propósito principal del protocolo LoRa es poder conseguir establecer conexiones de largo alcance. Estas comunicaciones están enfocadas para pequeños dispositivos que desean transmitir poca información, y por tanto, sin necesidad de obtener grandes velocidades. Una premisa en este tipo de dispositivos, es emplear el menor consumo de energía posible, para así dar un mayor tiempo de vida a las baterías.

La intención de LoRa es establecer una red de dispositivos conectados, a los que se les llama nodos. Estos nodos se comunican, mediante un enlace inalámbrico con un elemento de mayor potencia, que es capaz de comunicarse con todos estos elementos de manera gestionada. A este elemento se le conoce como Gateway [5]. Además de interconectar los nodos con una fuente de potencia más elevada, el Gateway debe ser capaz de conectarse por otro método de

red, con ancho de banda mayor, y poder transmitir toda esta información recopilada a aquellos dispositivos que lo requieran.

Se trata de un protocolo abierto, por lo que el número de desarrolladores y productos de mercado puede crecer de forma exponencial.

Sus características más importantes son:

- Protocolo escalable [6]
- Consumo muy reducido de energía.
- Facilidad de implementación.
- Alto rango de alcance.

LoRa tiene un rango de más de 15 kilómetros, y una capacidad de hasta 1 millon de nodos. La combinación de baja potencia y largo alcance hace que se limita la velocidad de datos máxima, dónde establece unos niveles cercanos a 50 kilobits por segundo (Kbps).

En lo que se refiere a la topología de red, una red LoraWAN clásica se compone de una serie de dispositivos finales que se conectan a Gateways, y éstos envían toda la información a un servidor. A través de una interfaz de programación de aplicaciones (API) la información se entrega a una aplicación final para el usuario.

En lo que se refiere a los nodos, en LoRa los nodos pueden funcionar con una conexión punto a punto, o conexión tipo “mesh”. La principal características de este modo es que no se requiere de un intermediario que administre la comunicación, sino que los dispositivos pueden enviarse información entre ellos directamente. En el modo “mesh” disponemos de un nodo que se encarga de coordinar la red, pero tiene una desventaja, y es que esta está limitada a 255 redes de 255 nodos, ya que el nodo coordinador únicamente puede escuchar un nodo a la vez. Este tipo de modo es muy útil para comunicaciones sencillas. Como veremos más adelante, en líneas futuras, el diseño de nuestra shield va destinado a la creación de un validador multiprotocolo, el cual permita validar la cobertura de señal de varios protocolos, y este modo de funcionamiento nos será realmente útil.

Por el contrario, en el modo LoRaWAN, los nodos se deben conectar forzosamente a un gateway que soporta hasta un máximo de 62500 nodos, y que puede escuchar hasta 8 nodos a la vez.

2.1.2 SigFox

El objetivo principal de SigFox consiste en permitir que todo objeto pueda conectarse a Internet, de forma asequible, dondequiera que se encuentre y sin necesitad de recargar su batería. La solución de conectividad SigFox se basa en una infraestructura de antenas y de estaciones base totalmente independientes de las redes ya existentes. Apuesta tecnológicamente por las redes LPWA [7], que ofrece una tecnología a un precio asequible y que permite desplegar dispositivos IoT de forma masiva.

A nivel tecnológico, SigFox utiliza la UNB [8], basada en una tecnología de radio, para conectar los dispositivos a su red. La utilización de la UNB es esencial para suministrar una red de alta capacidad, evolutiva, y de muy bajo consumo energético.

En SigFox se puede enviar entre 0 y 140 mensajes por día y cada mensaje puede contener hasta 12 bytes de datos reales de carga útil. El protocolo transmite el ID [9] del dispositivo, de modo que los 12 bytes representan la carga útil real y no existe ningún límite sobre cómo estructurar estos 12 bytes.

Las características principales de SigFox son:

- Tecnología de banda estrecha (Ultra Narrow Band).
- Bajo coste, tanto del dispositivo como del servicio.
- Muy eficientes en el uso de energía (pueden funcionar durante años a pilas).
- Gran cobertura.
- Requiere pocas estaciones base (miles de sensores pueden controlarse desde una misma estación).
- Excelente penetración bajo tierra, lo cual mejora la cobertura y amplía los usos.
- Sensibilidad de recepción de la señal relativamente baja en los terminales.
- Robustez del servicio ante interferencias de la señal.

En cuanto a la topología de red, SigFox trabaja con una topología de red en forma de estrella. Su funcionamiento es muy sencillo:

- Los objetos o nodos transmiten su mensaje a la red SigFox cuando la señal de radio alcanza las estaciones base dentro del campo de cobertura.
- Cada una de las estaciones base está conectada a la nube SigFox, gracias al tipo de enlace Punto-a-Punto.
- La nube transmite los mensajes a los servidores y a las plataformas IT de los clientes.

2.1.3NarrowBand IOT (NB IoT)

Narrow Band IoT (NB-IoT) es un nuevo estándar de comunicación del 3GPP [10] sobre espectro licenciado LTE [11]. Ofrece una cobertura adicional de hasta 20 dB sobre conexiones 2G [12]. Su consumo de energía es mínimo, permitiendo una duración de las baterías de más de 10 años. El tamaño de los dispositivos es mínimo, con un coste por dispositivo reducido. Además, se trata de un protocolo bidireccional [13], permitiendo usos en los que es necesario actualizar o gestionar el dispositivo de forma remota. Una de las características más importantes de NB-IoT es que tan solo necesita un ancho de banda de 200 KHz para soportar un gran número de usuarios, y como ya hemos mencionado, con un requerimiento de energía muy reducido. Podemos apreciar, a continuación, una tabla que nos muestras las ya mencionadas características de los protocolos descritos hasta el momento:

	LoRa	**Sigfox**	**NB-IoT**
Rango	1-3Km (entorno urbano) Hasta 15 Km entornos suburbanos	3-10 Km entorno urbano Hasta 30 Km entornos suburbano	3-15 km entorno urbano Hasta 35 Km entornos suburbanos
Topología de red	Ad-hoc, punto a punto, estrella o malla	Punto a punto, estrella	Ad-hoc, punto a punto
Frecuencia Operación	433-868 MHZ (Europa)	868 MHz y 902 MHz	Bandas LTE
Consumo de potencia	Bajo	Bajo	Medio
Licencia espectro	No licenciado	No licenciado	Licenciado
Velocidad de transmisión	Hasta 50kbit/s	Hasta 200 Kbit/s	Hasta 250 kbit/s

Tabla 1: Características de los protocolos LoRa, SigFox y NB-IoT

2.1.4 Bluetooth

Bluetooth es un protocolo de comunicación inalámbrico desarrollado originalmente por *Ericsson [14]* que opera sobre frecuencia sin licencia [15]. Se concibió para redes PAN [16] con el propósito de transferir datos de manera inalámbrica. Sin embargo, presenta limitaciones intrínsecas de diseño: su consumo de energía es elevado (no está optimizado para dispositivo que funcionen a pilas), y su alcance no suele superar el de la habitación donde se encuentren los dispositivos.

Bluetooth tiene un radio de acción de 10 a 100 metros, dependiendo de la clase del dispositivo:

- 1mW para 10 metros,
- 100mW para 100 metros.

En lo que se refiere al tipo de topología, Bluetooth utiliza las piconets. Una piconet es una red informática cuyos nodos se conectan utilizando Bluetooth. Una piconet puede tener un máximo de siete dispositivos. Además, en una piconet disponemos siempre de un dispositivo que hará de “master”, mientras que el resto harán de “esclavos”.

Hace unos años, Nokia [17] desarrolló una nueva evolución del estándar Bluetooth para superar estas limitaciones llamado Bluetooth Low-Energy. Bluetooth LE se fundamenta en la reducción del consumo y, por tanto, en minimizar la potencia de transmisión de la señal radio utilizada y el radio de cobertura. Con estas premisas, llegamos a tener una velocidad de conexión de hasta 1 Mbps en la capa física. Además permite la comunicación entre dispositivos utilizando la banda de frecuencia de 2.4 GHz.

2.1.5 WiFi HaLow

Se trata de la nueva especificación del protocolo de comunicación WiFi [18], el cual permite trabajar en la banda de frecuencia de los 900 MHz, y de esta manera permite una mejor cobertura y alcance en interiores, concretamente, permite duplicar el alcance utilizando menos potencia. Al trabajar en esta frecuencia, no mejoramos las velocidades de transmisión, pero podemos controlar mejor las interferencias y las pérdidas ocasionadas por obstáculos.

Además se caracteriza por un protocolo de comunicación de bajo consumo. El alcance en interiores es el doble que una conexión WiFi actual, y además es más robusta, capaz de lidiar con paredes y con barreras físicas, y sin renunciar a la actual seguridad que proporcionan las redes WiFi.

En este nuevo estándar los anchos de banda de los canales utilizados son de 1 MHz y 2 MHz en la mayoría de los casos, y de 4, 8 y 16MHz para aquellos dispositivos que necesiten una mayor velocidad.

En lo que se refiere a la topología, WiFi HaLow se dispone de dos métodos de funcionamiento. Trabaja en modo infraestructura, y en modo “Ad Hoc”. En modo infraestructura se requiere de un punto de acceso conectado a la red, y a partir de aquí se pueden conectar los routers que conformaran la red. En el modo “Ad Hoc”, no se requiere un punto de acceso, es decir, los dispositivos se conectan entre ellos sin necesidad de tener un punto de acceso. Además, existe una variante ya conocida también, y es que pueden actuar en modo mesh, que como ya sabemos, cada elemento de la red se comporta como un nodo, capaz de encaminar los paquetes, y comunicándose directamente entre los dispositivos.

2.1.6 ZigBee

ZigBee es un conjunto de protocolos de alto nivel de comunicación. Se utiliza para la radiodifusión digital de datos, buscando ahorrar lo máximo posible en energía. Se trata de una tecnología basada en el estándar de la IEEE [19], el IEEE 802.15.4.

La tecnología de comunicación inalámbrica ZigBee adopta por lo general la banda de 2.4GHz para comunicarse con el resto de dispositivos, dividido en 16 canales, y cada uno de ellos con un ancho de banda de 5MHz. Además utiliza el protocolo CSMA/CA [20] para evitar colisiones durante la transmisión.

Una de las principales ventajas de ZigBee es lo sencillo y el bajo coste que supone producir dispositivos con esta tecnología de comunicación. Se trata de una tecnología mucho más sencilla que Bluetooth, por ejemplo.

En los que se refiere al tipo de topología, en ZigBee existen tres tipos de topología: en estrella, en árbol y en red mallada (mesh). En las tres topologías siempre hay un nodo de red que asume el papel de coordinador central, y este se encarga de centralizar la adquisición y las rutas de comunicación entre los dispositivos conectados a la red.

En la topología en estrella, todos los dispositivos de la red únicamente pueden comunicarse con el coordinador.

En la topología de árbol, un coordinador de ZigBee establece la red inicial. A partir de aquí, los routers ZigBee son los encargados de formar las ramas y transmitir los mensajes.

En la topología de malla, cada dispositivo puede comunicarse directamente con cualquier otro dispositivo, siempre y cuando dos o más dispositivos se encuentren lo suficientemente cerca para establecer una comunicación exitosa.

A continuación podemos ver una tabla que muestra las características de los protocolos de comunicación ZigBee, Bluetooth y WiFi, presentados previamente:

	ZigBee	Wi-Fi HaLow	Bluetooth
Rango	10-100 metros	50-100 metros	10-100 metros
Topología de red	Ad-hoc, punto a punto, estrella o malla	Punto a punto, Ad-hoc	Ad-hoc, redes pequeñas
Frecuencia Operación	868 MHZ (Europa) 900-928 MHZ (NA)	900 MHz, y mantiene compatibilidad con 2.4 y 5 GHz	2.4 GHz
Consumo de potencia	Muy bajo	Bajo	Bajo
Licencia espectro	No licenciado	No licenciado	No licenciado
Velocidad de transmisión	250 kbit/s	Por determinar	Hasta 3000 kbit/s

Tabla 2: Características de los protocolos ZigBee, Bluetooth y WiFi

2.1.7 LTE-M

El LTE categoría M está estandarizado por la 3GPP, y busca reemplazar la red 2G, que utilizan los dispositivos para comunicaciones M2M [21].

El protocolo LTE-M, además, ofrece unas características técnicas ligeramente superiores a NB-IoT, en términos de latencia [22] y ancho de banda. El consumo energético de estos chips, pese a ser muy inferior al del LTE estándar, supera al de los módems diseñados para la red NB-IoT, por tanto consumen más que los que emplean el protocolo NB-IoT.

Tanto NB-IoT como LTE-M operan sobre bandas de espectro licenciado en conexiones LTE. La razón que lleva a las compañías a operar en estas bandas de frecuencia tan bajas son las siguientes: mejor cobertura en interiores y el uso de antenas con un mayor alcance.

LTE-M soporta velocidades de subida y bajada de 375 kbps. La vida útil de la batería dura hasta 10 años con una sola carga en algunos casos de uso. También contribuye a reducir los costes de mantenimiento de los dispositivos desplegados, incluso en lugares donde los dispositivos no se pueden conectar directamente a la red eléctrica.

2.1.8 Nordic Semiconductor

Nordic Semiconductor está comprometido en proporcionar soluciones de alta calidad, fáciles de usar, confiables y rentables de ultra baja potencia al mercado inalámbrico. Puede trabajar con diferentes líneas de productos. Actualmente, ofrece cuatro líneas de productos:

- **RF 2.4GHz**: Solución propia de Nordic Semiconductor. Trabaja en la banda de libre uso de 2.4GHZ, y el módulo transceptor que vamos a elegir trabajará con esta solución.
- **ANT**: es un protocolo de red de ultra baja potencia, flexible y fácil de usar para la comunicación en banda ISM de 2,4 GHz. ANT proporciona interoperabilidad a través de la red gestionada ANT+
- **Bluetooth de baja energía**.
- **Sub_1GHz**: Solución propia de nordic semiconductor. Trabaja en la banda ISM de 433/868/915MHz. Se trata de dispositivos de baja potencia, bajo consumo y facilidad para su integración.

En lo que se refiere a topología de red, y como ya hemos comentado, existen tres tipos de topología de red: estrella, árbol, o de malla (mesh), y cuyo funcionamiento ya hemos explicado.

Los protocolos que se acaban de presentar son los protocolos más utilizados y con más reputación dentro del mundo del IoT, y como podemos ver, todos tienen características bastante similares, buscando un compromiso de muy baja potencia y que el coste sea reducido.

Después de esta presentación, nos adentramos en los productos que ofrece actualmente el mercado para trabajar con cada uno de los protocolos descritos, ya que nuestra idea es realizar una shield multiprotocolo, que permita ensamblar a la vez diferentes chips. Es por ello que vamos a realizar un estudio de mercado para determinar cuáles van a ser los chips, que por sus características, y por nuestros requerimientos del sistema, se adapten mejor a nuestra shield.

2.2 Estudio de mercado

Puesto que nuestro objetivo es trabajar con varios de los protocolos aquí mencionados, vamos a realizar una búsqueda exhaustiva para ver que opciones ofrece el mercado actual, en lo que a chips se refiere. Es importante recalcar, que se trata de una estimación de algunos de los muchos chips que tenemos actualmente en el mercado.

2.2.1 Protocolo de comunicación LoRa

	Alcance	Voltaje	Velocidad Datos	Pot Tx	Program.	Rango Frecuencia	Precio	Consumo Rx.
RN 2483	15 Km sub. 5 Km urban	2.1-3.6V	50 Kbps	14 dBm	SI	433/868 MHz	10.74€	14 mA
RF-LORA-868-SO	16 KM	1.8-3.6V	50Kbps	13 dBm	SI	868 MHz	19.91€	10mA
RN2903	15 Km max	2.1-3.6V	50 Kbps	14 dBm	SI	902-928 MHz	11.40€	13.5mA

Tabla 3: Chips protocolo LoRa

2.2.2 Protocolo de comunicación SigFox

	Alcance	Voltaje	Velocidad Datos	Pot Tx	Program.	Rango Frecuencia	Precio	Consumo Rx.
ATA 8520	1 Km max	1.9-3.6V	100bps	14.5 dBm	SI	868-868.6 MHz	10.74€	10.4 mA
FiPy	1 Km aprox	3.3-5.5V	300 Kbps DL 375 Kbps UP	13-15 dBm	SI	699 to 2690 MHz 4MB RAM 8 MB FLASH	54€	17 mA
SiPy RCZ1 & RCZ3	1Km aprox	3.3-5.5V	300 Kbps DL 375 Kbps UP	14-15 dBm	SI	868 MHz 512KB RAM	33.79€	17mA

Tabla 4: Chips protocolo SigFox

2.2.3 NB IoT

	Alcance	Voltaje	Velocidad Datos	Pot Tx	Program.	Rango Frecuencia	Precio	Consumo Rx
SARA-N2	16 KM	1.8-3.6V	27.2Kbps DL – 62.5Kbps UP	23 dBm	SI	880 a 915 MHz	41.40€	10mA
NANO-S100	15 Km max	2.2-5.5V	0.5-200Kbps DL 0.3-180 Kbps UL	14 dBm	SI	2.4 GHz	10.35€	12 mA

Tabla 5: Chips protocolo NB-IoT

2.2.4 Bluetooth

	Alcance	Voltaje	Velocidad Datos	Pot Tx	Program.	Rango Frecuencia	Precio	Consumo Rx
ATBTLC 1000	50 m aprox	1.8-4.3V	200Kbps	10 dBm	SI	2.4 GHz 128 KB RAM 128 KB ROM	5.35€	4 mA
ATSAM B11	50 m aprox	2.3-3.6V	200Kbps	8 dBm	SI	2.4 GHz 128 KB RAM 256KB flash	2.95€	4.5mA
IS1678	50 m aprox	3.2.-4.3V	200Kbps	10 dBm	SI	2.4 GHz	2.27€	13.5mA

Tabla 6: Chips protocolo Bluetooth

2.2.5 WiFi-HALow

	Alcance	Voltaje	Velocidad Datos	Pot Tx	Program.	Rango Frecuencia	Precio	Consumo Rx
ATWINC 1500	80-150 m aprox	3-4.2V	6Mbps	17 dBm	SI	2.4 GHz	9.95€	60 mA
RN1810	80-150 m aprox	3.15-3.45V	6Mbps	18dBm	SI	2.4 GHz	14.25€	46 mA
ATWILC 1000	80-150 m aprox	2.5-4.2V	6Mbps	18 dBm	SI	2.4 GHz	9.03€	63 mA

Tabla 7: Chips protocolo WiFi-HALow

2.2.6 LTE-M

	Alcance	Voltaje	Velocidad Datos	Pot Tx	Progra m.	Rango Frecuencia	Precio	Consumo Rx
GPY	1 Km aprox	3.3-5.5V	1 Mbps	15 dBm	SI	699 a 2690 MHz 4MB RAM 8 MB FLASH	44€	330 mA
FiPy	1 Km aprox	3.3-5.5V	300 Kbps DL 375 Kbps UP	13-15 dBm	SI	699 a 2690 MHz 4MB RAM 8 MB FLASH	54€	17 mA

Tabla 8: Chips protocolo LTE-M

2.2.7 Nordic Semiconductor

	Alcance	Voltaje	Velocidad Datos	Pot Tx	Progra m.	Rango de frecuencia	Precio	Consumo Rx
NRF24L01	1 Km aprox	3.3-5.5V	2000 Kbps	0 dBm	SI	699 a 2690 MHz 4MB RAM 8 MB FLASH	3.57€	330 mA

Tabla 9: Chip protocolo Nordic Semiconductor

2.2.8 ZigBee

	Alcance	Voltaje	Velocidad Datos	Pot Tx	Program.	Rango Frecuencia	Precio	Consumo Rx
XB24-AWI-001	30-90 m aprox	2.8-3.4V	250Kbps	10 dBm	SI	2.4 GHz	26.10€	50 mA
XBP24-AUI-001	30-100 m aprox	2.8-3.4V	250Kbps	10 dBm	SI	2.4 GHz (12 canales)	35.09€	50 mA
XB8-DMUS-002	30-100 m aprox	2.7-3.6V	250Kbps	10 dBm	SI	863-870 MHz	23.39€	27 mA
XBP24B ZWIT	30-90m	2.8-3.4V	250Kbps	17 dBm	SI	2.4GHz	20 €	50mA

Tabla 10: Chips protocolo ZigBee

El siguiente paso, una vez disponemos de toda la información, es estudiar que protocolos vamos a querer utilizar, ya que necesitamos saber que chips se van a ensamblar en nuestra placa.

2.3 Elección Protocolos

Después del estudio de los protocolos existentes en el mundo del Internet de las cosas, debemos seleccionar que protocolos vamos a utilizar para nuestro proyecto, y decidimos finalmente, que basaremos nuestro proyecto en los protocolos más extendidos y con más uso actualmente, y que, una característica en común, es que todos ellos disponen de que su uso de espectro no necesita licencia. Muchos de los protocolos disponen de características muy similares, pero debido a que tenemos restricción de costes, optamos por aquellos que, además de lo ya mencionado, disponen de chips con un coste bastante asequible.

Por otro lado, la Raspberry Pi dispone de tecnología WiFi y Bluetooth, por lo que nos aprovechamos de esta situación, y de esta manera, podemos ahorrarnos la compra de chips que funcionen con los protocolos WiFi y Bluetooth, además del coste adicional que supondría la implementación de éstos en la shield.

Los protocolos de comunicación seleccionados que vamos a implementar son:

- Zigbee
- Bluetooth
- Nordic Semiconductor
- LoRa
- WiFi

Una vez seleccionados los protocolos que vamos a utilizar, debemos decidir cuáles son los chips que vamos a elegir. Como ya se ha visto, hemos hecho un estudio de mercado con las diferentes opciones que tenemos hoy en día. Actualmente hay una gran variedad de chips para cada uno de los protocolos aquí mencionados, así que debemos basarnos en algunos aspectos muy importantes, relacionados con el diseño de la placa. Los aspectos más importantes que hacen que nos podamos decantar por uno u otro chip son los siguientes:

- En primer lugar, queremos chips que tengamos al alcance, que su compra esté disponible en nuestro territorio. Nos hemos encontrado con algunos chips que son difíciles de obtener, o bien por su rareza, o porque ya no se distribuyen.
- Además, puesto que el proyecto es un Spin-off de un proyecto más grande que está realizando la compañía **AEInnova** [23], hemos pensado que sería interesante reaprovechar algunos chips que tienen en stock, y que son perfectamente válidos para la realización del proyecto.
- Debemos tener muy en cuenta las características del chip. Como podemos ver en las tablas de los componentes, necesitamos chips que sean programables, que su coste sea lo más reducido posible, siempre cumpliendo con los requisitos del sistema. Una de las partes más importantes, y más influyentes, es el rango de alimentación de cada chip. Disponemos de una Raspberry PI, y sabemos que puede alimentar a 3.3V o a 5V, pero que todas sus salidas GPIO se alimentan a 3.3V. Es por ello que buscamos también, para hacer el diseño más sencillo, chips que funcionen con un rango de alimentación cercano a 3.3V.

Después de analizar todas las posibilidades de las que disponemos, hemos decidido de forma definitiva, que los chips elegidos deben ser programables, con un coste medio reducido y que se alimenten con un rango cercano a los 3.3V que nos proporciona la Raspberry Pi.

2.3.1 Elección de los chips

Una vez determinados los protocolos con los que vamos a trabajar, determinamos también los chips que vamos a utilizar. En este caso, y aprovechándonos de los recursos de los que disponemos, y que entran dentro de las opciones que hemos estado barajando, los chips para cada protocolo son:

- LoRa → RN2483
- Nordic Semiconductor → NRF24L01
- ZigBee → XBP24BZWIT

Cabe mencionar que, a pesar de haber hecho un estudio de mercado del protocolo Bluetooth, y de los chips que funcionan con este protocolo, finalmente decidimos no colocar ningún chip supletorio del protocolo Bluetooth.

El motivo de tal decisión es porque el modelo de Raspberry PI que vamos a utilizar ya dispone de conexión Bluetooth. Gracias a ello, podemos ahorrarnos el coste de un chip BT, además del tiempo que conlleva su diseño y su implementación en nuestra PCB [24], así que decidimos aprovechar uno de los recursos que nos ofrece nuestra Raspberry PI.

Igual que pasa con Bluetooth, la Raspberry Pi dispone de un adaptador de red WiFi, por lo que decidimos que es innecesario invertir en un chip que pueda hacer la misma función.

2.4. Conclusiones sobre el estado del arte

En este capítulo del trabajo se han explicado los protocolos más relevantes que existen actualmente. Además, hemos decidido qué protocolos vamos a utilizar, y que chips vamos a utilizar.
Nuestro proyecto estará formado por los protocolos Bluetooth, WiFi, ZigBee, LoRa y el propio protocolo de Nordic Semiconductor.
De ahora en adelante, pasaremos de la teoría y del estudio de todos estos componentes, a la parte práctica, dónde vamos a poner en escena todo lo ya mencionado.

3. Planificación del proyecto

Para poder ver de una forma muy gráfica y clara como se han invertido el trabajo en el estudio y el desarrollo de este proyecto, vamos a recurrir a una realizar un diagrama de Gantt. En este diagrama podemos ver la distribución del trabajo que se ha hecho en el transcurso del tiempo, definiendo de forma aproximada las horas invertidas en cada una de las tareas que han conformado el proyecto, desde sus inicios hasta el final.

Para la realización del diagrama, hemos dividido las tareas del proyecto en 4 fases:

- Fase 1: Estudio del escenario.
- Fase 2: Determinación de los componentes para el diseño de la placa.
- Fase 3: Diseño de la placa.
- Fase 4: Pruebas / Test.
- Fase 5: Redacción memoria.
- Fase 6: Presentación del proyecto

Cada una de las fases está subdivida con una generalización de las tareas que se han realizado en cada una de estas.

3.1 Fase 1: Estudio del escenario

En esta primera fase, tomamos como primer evento la asignación del proyecto. Pese a que el proyecto se comenzó a realizar a partir de enero, es evidente que el momento de la asignación debe estar presente. Posteriormente, realizamos una primera toma de control para definir las bases del proyecto:

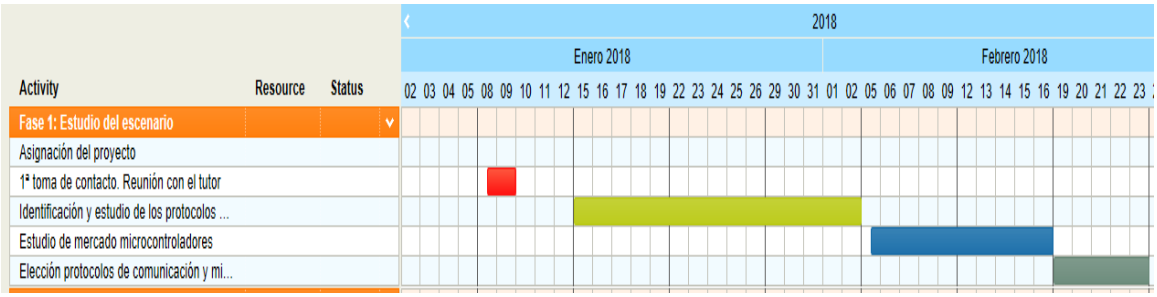

Figura 2: Fase 1 Diagrama de Gantt

3.2 Fase 2: Estudio de los componentes para el diseño del circuito impreso

En esta segunda fase, y después de una primera toma de contacto con los protocolos y de los componentes que podemos utilizar, profundizamos en el funcionamiento de los protocolos elegidos, y del funcionamiento de los chips que elegiremos, procurando comprenderlos:

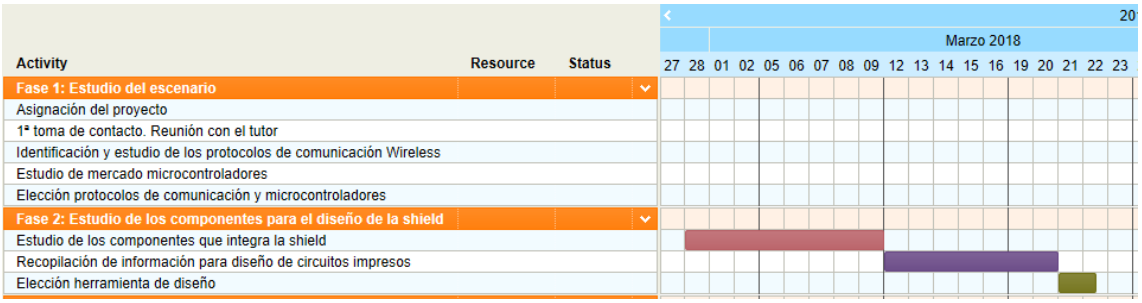

Figura 3: Fase 2 Diagrama de Gantt

3.3 Fase 3: Diseño del circuito impreso

Se trata de la fase principal del proyecto, donde todo lo estudio y compresión de la información procuramos aplicarla para diseñar la shield. Se trata de la fase más larga, y más compleja, puesto que, como ya se ha comentado en otras ocasiones, es un proceso muy costoso e iterativo hasta poder llegar al diseño final:

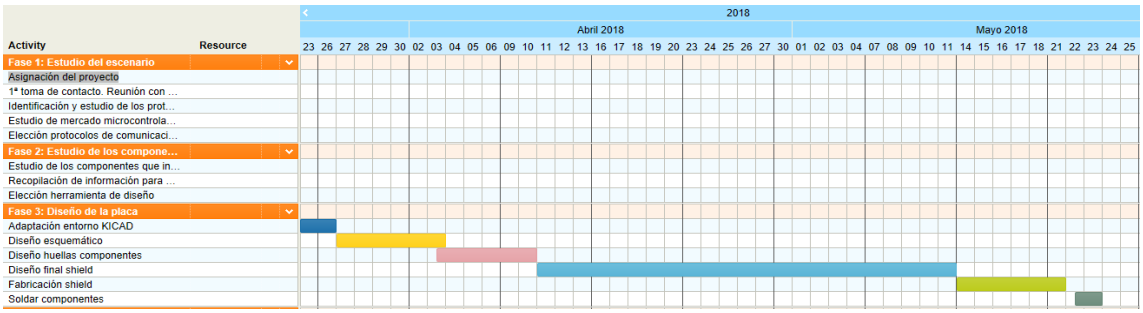

Figura 4: Fase 3 Diagrama de Gantt

3.4 Fase 4: Pruebas / Test

Se trata de la fase de pruebas de la placa, con el fin de poder demostrar que funciona correctamente, y que tiene un largo porvenir en el transcurso de la continuación de este proyecto, donde deberá ser utilizada para su programación:

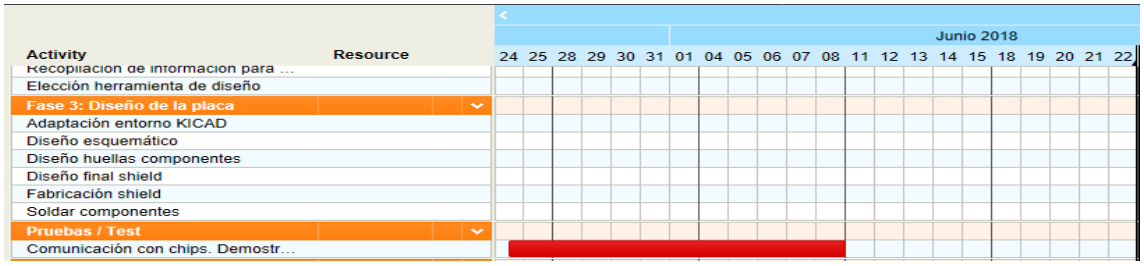

Figura 5: Fase 4 Diagrama de Gantt

3.5 Fase 5: Redacción – Presentación proyecto

Fase final del proyecto, en la que nos dedicamos a redactar la memoria que se entregará, y finalmente añadimos el día de la entrega, y la semana de presentación del proyecto ante el tribunal que se le asigne:

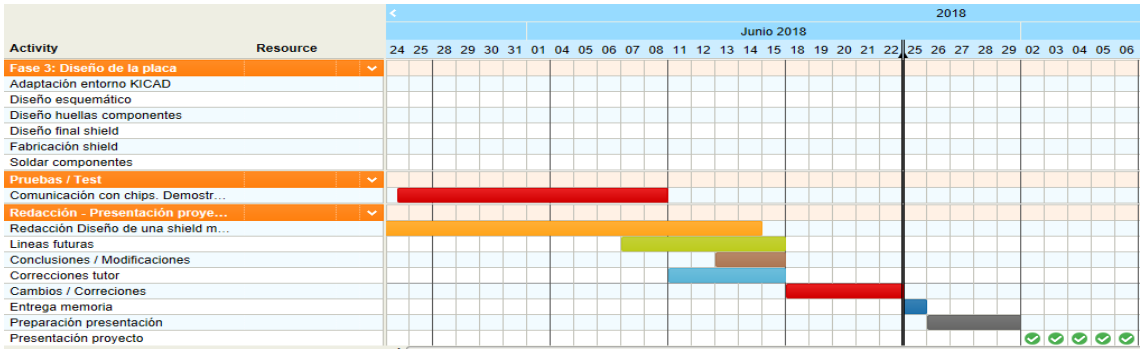

Figura 6: Fase 5 Diagrama de Gantt

4. Introducción a la herramienta de diseño de PCB KICAD

KICAD es una de las múltiples herramientas de diseño Hardware de libre distribución de las que disponemos actualmente. También existen otras herramientas, como por ejemplo Eagle, Multisim, Proteus, OrCad, PCB Wizard, etc.

Elegimos KICAD por su sencillez para trabajar, por la gran cantidad de tutoriales que existen, y porque dispone de una librería muy extensa en lo que a huellas se refiere, además de que se trata de una herramienta flexible y adaptable.

KICAD se estructura en 5 bloque:

- KICAD -> Administrador de proyectos.
- KICAD -> **Eeschema**: editor de esquemáticos.
- KICAD -> **cvPCB**: Seleccionador de huellas.
- KICAD -> **pcbNEW**: Entorno de diseño de los circuitos impresos (PCB).
- KICAD -> **Gerbview**: Visualizador de archivos Gerber.

Después de una pequeña presentación de la herramienta que vamos a utilizar para el diseño de nuestro proyecto, vamos a listar los componentes que tenemos que utilizar para poder realizar el esquemático.

4.1 Determinación de los componentes

Vamos a listar los componentes que debemos implementar en nuestra shield:

- **RN 2483**: chip que trabaja con el protocolo de comunicación LoRa.
- **NRF24l01**: chip que trabaja con el protocolo de comunicación Nordic.
- **XBee P24BZ**: chip que trabaja con el protocolo de comunicación ZigBee.

En esta primera fase, hemos designado los 3 módulos que utilizaremos para cada uno de los protocolos seleccionados. Cabe volver a recordar que no hay un chip para el protocolo Bluetooth ni para WiFi, ya que aprovechamos los módulos que incorpora la Raspberry PI.

4.1.1 Raspberry PI 3 model B +

¿Qué es una Raspberry PI? La Raspberry PI es una diminuta placa base, cuyas dimensiones son de 85 mm X 54 mm, en la cual se dispone de un procesador BroadCom BCM2837BO (ARMv8) a 1.4 GHz. La Raspberry Pi se alimenta a través de una conexión USB, la cual proporciona aproximadamente 5V. Pero todos los demás puertos GPIO de la Raspberry funcionan a 3.3V, y es un dato muy relevante de cara a nuestro diseño.

A continuación podemos ver la distribución de conexiones GPIO que proporciona la Raspberry PI:

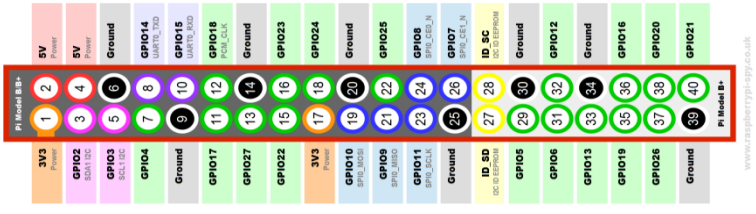

Figura 7: Distribución de los pines GPIO de la Raspberry Pi

Podemos ver que la Raspberry dispone de un conjunto de Pines que permite interactuar con ella. Hay una serie de pines, los cuales están diseñados para poder controlar circuitos electrónicos. Estos pines los podemos configurar y controlar directamente mediante el uso de lenguajes de programación como C o Phyton, ya que cuentan con extensas librerías para el control de los pines GPIO.

Debido al diseño de los chips que hemos elegido, disponemos de dos tipos de conexiones diferentes:

- En primer lugar, el módulo NRF24l01 utiliza una conexión de entrada SPI, es decir, va conectada a los pines GPIO10 Y GPIO9, MOSI y MISO respectivamente.
- Por otro lado, disponemos de los módulos RN 2483 y Xbee P24Bz, cuyas conexiones son conexiones UART, GPIO 14 (Tx) y GPIO 15 (Rx).

En lo que se refiere a la conectividad inalámbrica de la Raspberry, el modelo que vamos a utilizar dispone de conexiones tanto por Bluetooth como por WiFi. Respecto a otros modelos anteriores, en este modelo dispone de las siguientes características:

	Frecuencia	Estándar
Bluetooth	2.4GHz	IEEE 802.11.b/g/n/ac
WiFi	2.4GHz / 5GHz	Bluetooth 4.2, BLE

Tabla 11: Características conexiones inalámbricas Raspberry PI 3 B+

Además de los puertos GPIO, la Raspberry también tiene los siguientes puertos de conexión:

- GPIO 40 pines.
- HDMI.
- x USB 2.0.
- CSI (cámara Raspberry Pi).
- DSI (pantalla táctil).
- Toma auriculares / vídeo compuesto.
- Micro SD.
- Micro USB (alimentación).
- Power-over-Ethernet (PoE).

Por último, y no menos importante, tiene un precio de mercado cercano a los 40 euros. Se trata de una placa con un coste muy bajo en comparación a sus prestaciones, ya que con este último modelo, y gracias al nuevo procesador, ofrece una cantidad muy elevada de opciones para su desarrollo, no únicamente para el desarrollo de este tipo de aplicaciones.

Después de la presentación de la Raspberry PI, vamos a explicar que son las conexiones UART, las conexiones SPI y cómo funcionan.

4.1.1.1Estándar SPI

SPI (Serial Peripheral Interface): Estándar de comunicaciones, normalmente utilizado para la transferencia de información entre circuitos integrados en equipos electrónicos. Se trata de un estándar que permite controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj (comunicación sincrónica).

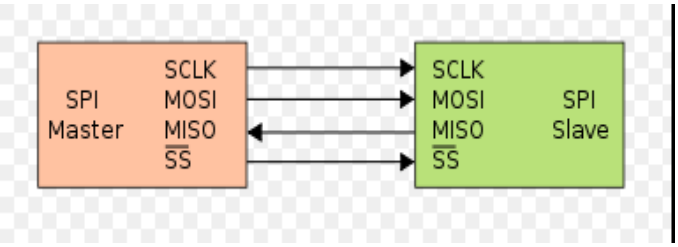

Figura 8: Conexión protocolo SPI

El protocolo SPI es un **protocolo síncrono**. La sincronización y la transmisión de datos se realiza a través de 4 señales:

- **SCLK (Clock)**: Es el pulso que marca la sincronización. Con cada pulso de este reloj, se lee o se envía un bit.
- **MOSI (Master Output Slave Input)**: Salida de datos del Master y entrada de datos al Slave.
- **MISO (Master Input Slave Output)**: Salida de datos del Slave y entrada al Master.
- **SS/Select**: Para seleccionar un Slave, o para que el Master le diga al Slave que se active.

Aquí podemos apreciar cuales son las principales ventajas y desventajas de este estándar:

Ventajas:

- Comunicación Full Duplex.
  □2□ o SMBus
- Se trata de un protocolo muy flexible, por lo que podemos tener un control absoluto de todos los bits transmitidos.

Desventajas:

- No permite fácilmente tener varios servidores conectados al bus.
- No hay control de flujo por Hardware.
- No tenemos señal de asentimiento, es decir, el servidor podría estar enviando información sin que estuviese conectado ningún cliente y no se daría cuenta de nada.
- Funciona a distancias cortas.

4.1.1.2 Estándar UART

UART, son las siglas en inglés de *Transmisor-Receptor Asíncrono Universal*. Se trata del dispositivo que controla los puertos y dispositivos serie, y utiliza el estándar serie RS232.

El controlador del UART es el componente clave del subsistema de comunicaciones serie en la Raspberry PI. El UART coge bytes de datos, y transmite los bits de forma individual, y de forma secuencial. De la misma forma, en destino, el segundo UART reensambla los bits en bytes completos.

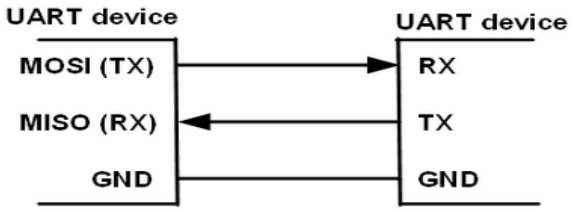

Figura 9: Conexión estándar UART

Como se puede apreciar, el pin TX de un dispositivo está conectado al RX del otro dispositivo, y viceversa, y ambas masas están conectadas.

Se trata de un bus muy utilizado para la programación de chips de sistemas integrados con un ordenador, debido a que no hace falta que los chips traigan un bootloader [25] precargado.

Ahora que ya sabemos qué tipo de conexiones vamos a utilizar para el diseño de la placa, el siguiente paso es encontrar solución al siguiente problema: nos encontramos con que la Raspberry solo proporciona un único puerto UART (UART TX, UART RX), mientras que disponemos de dos chips cuyo funcionamiento es a través de este estándar.

Es por ello que debemos plantearnos el uso de un multiplexor, y un demultiplexor. Vamos a recordar brevemente que es un multiplexor y un demultiplexor.

4.1.2 Multiplexor

Los multiplexores son circuitos combinacionales, que disponen de varios canales de entrada y solamente un canal de salida. Solamente un canal de la entrada pasará a la salida, el cual será escogido por una señal de control.

He aquí un esquema:

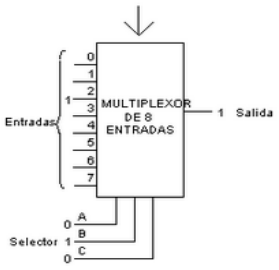

Figura 10: Ejemplo de multiplexor

4.1.3 Demultiplexor

Los demultiplexores son circuitos que disponen de una entrada de información de datos 2□, por la que ha de salir el dato que presente la entrada.

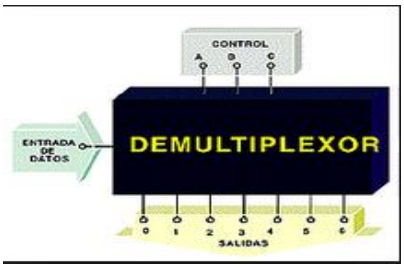

Figura 11: Ejemplo demultiplexor

Ahora que sabemos cómo funciona un multiplexor, y un demultiplexor, vamos a exponer la situación que tenemos:

- Como ya he dicho, disponemos de únicamente 2 puertos UART (UART Tx, UART Rx).
- Disponemos de 2 chips cuyos puertos de transmisión y recepción funcionan con este sistema.
- Es por ello que necesitamos un multiplexor, de forma que cada conexión de salida de los chips hacia nuestra Raspberry, es decir, necesitamos un multiplexor de N entradas, y una única salida, regulada por una señal de control.
- De modo inverso, cuando tenemos una comunicación cuya dirección va desde la Raspberry hacia a los chips. Es por ello que disponemos de una única entrada, y en función del chip con el que nos queramos comunicar, elegiremos una u otra salida, por lo tanto, utilizaremos un demultiplexor.

Por lo tanto, nuestro siguiente objetivo es buscar un multiplexor y un demultiplexor adecuados a los requerimientos de nuestro sistema. Después de buscar en el mercado, del que disponemos de múltiples opciones, decidimos utilizar un módulo que puede actuar como MUX/DEMUX. El módulo en cuestión es el 74LVC1G3157.

4.1.4 74LVC1G3157GV

El módulo 74LVC1G3157GV es un componente fabricado por Nexperia [26], y puede actuar tanto de multiplexor como demultiplexor. Se trata de un componente que dispone de un rango de alimentación que va desde 1.65V hasta un máximo de 5.5V. Dispone de una entrada que actúa como selector digital (S), dos entradas / salidas independientes (Yo / Y1), y una entrada/salida común (Z). Aquí podemos ver su diagrama:

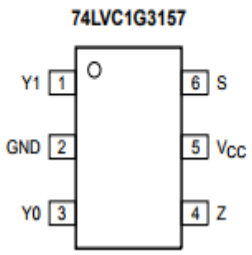

Figura 12: Configuración pines 74LVC1G3157

Y aquí podemos apreciar su diagrama lógico:

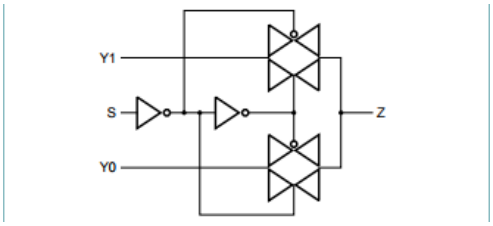

Figura 13: Diagrama lógico 74LVC1G3157

Es importante hacer mención de porqué se elige este modelo en concreto, y es por la sencilla razón de que necesitamos un chip con el encapsulado lo más grande posible, es decir, que tenga las dimensiones lo más grandes posibles dentro de todas las posibilidades que ofrezca el fabricante, ya que posteriormente seremos nosotros quien soldaremos todos los componentes de la placa, y si disponemos de componentes muy pequeños es muy probable que no sea posible soldarlos, o que nos resulte complicado.

4.1.5 Convertidor de nivel lógico Bidireccional

Prosiguiendo con lo antes mencionado, puesto que vamos a utilizar dos módulos cuyo voltaje de alimentación es de 5V, y éstos deben ir conectados a la Raspberry, y a los chips, cuyo voltaje de alimentación es de 3.3V, debemos implementar un convertidor lógico bidireccional, es decir, un elemento que nos permita pasar de un voltaje a otro, y viceversa, para evitar el mal funcionamiento de los componentes. Si disponemos de un componente cuyo voltaje de funcionamiento es de 3.3V y le introducimos datos de otro dispositivo que funciona a 5V, lo único que podemos conseguir es quemar el chip. Es por ello que vamos a utilizarlos.

Como ya es costumbre, disponemos de una cantidad importante de elementos de conversión en el mercado. Y es por ello, que nos decidimos por el convertidor de Sparkfun [27], que vamos a explicar a continuación.

El convertidor de nivel lógico bidireccional Sparkfun es un dispositivo que convierte de forma segura las señales de 5V a 3.3V y de 3.3V a 5V al mismo tiempo. Además, se puede establecer con éxito sus altos y bajos voltajes y subir y bajar entre ellos de manera segura en el mismo canal. Cada convertidor de nivel tiene la capacidad de convertir 4 pines en la parte alta, y de 4 pines en la parte baja, con dos entradas y dos salidas provistas en cada lado.

Su esquema es el siguiente:

Figura 14: Esquema convertidor lógico bidireccional

Y, de forma aún más gráfica, podemos observar la simplicidad de su funcionamiento:

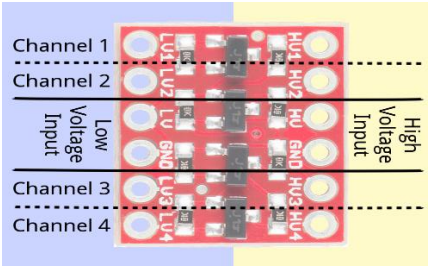

Figura 15: Canales de un convertidor lógico digital

Por lo tanto, una vez presentado el convertidor, volvemos a replantear nuestro esquema.

Necesitamos un multiplexor, que conecte el GPIO Tx de la Raspberry con los módulos RN2483 y Xbee24b. Como ya hemos dicho, nuestro MUX tiene un voltaje de alimentación de 5V, por lo que necesitamos dichos convertidores, para así, adaptar los flujos de información al voltaje pertinente de cada dispositivo.

Y por último, vamos a presentar los dispositivos que permitirán trabajar con los diferentes protocolos de comunicación.

4.1.6 Descripción módulo RN 2483

El RN2483 es un módulo transceptor de tecnología LoRa de largo alcance y baja potencia. El módulo RN2483 cumple con las especificaciones del protocolo LoRaWAN. Integra RF, un controlador de banda base y procesador de comandos de interfaz de programación de aplicaciones, lo que lo convierte en una solución completa de largo alcance. Sus características principales son las siguientes:

- Rango de funcionamiento de 2,1V a 3,6V.
- Potencia de transmisión ajustable de hasta 14dBm.
- Opera en las bandas de frecuencia entre 433MHz y 868MHz.
- Tasa de bits de RF programable de hasta 300Kbps con modulación FSK.
- Rango de temperatura entre -40ºC y 85ºC.
- 14 GPIO’s para control y estado.
- Corriente de alimentación de hasta 38,9 mA en transmisión.

Aquí podemos apreciar su diagrama de pines, para ver su diseño:

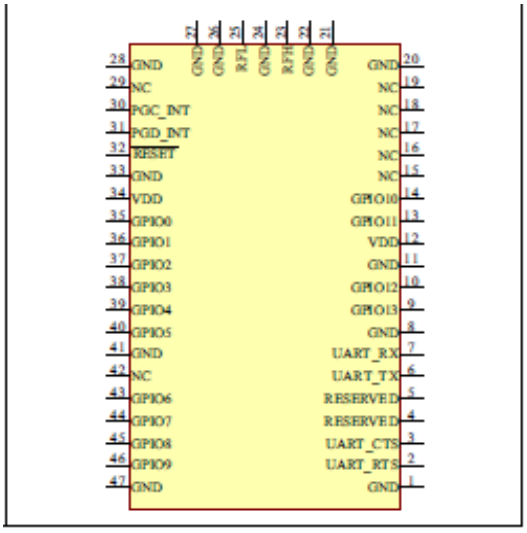

Figura 16: Diagrama de pines del chip RN2483

Y, a continuación, podemos ver el diagrama de bloques del chip:

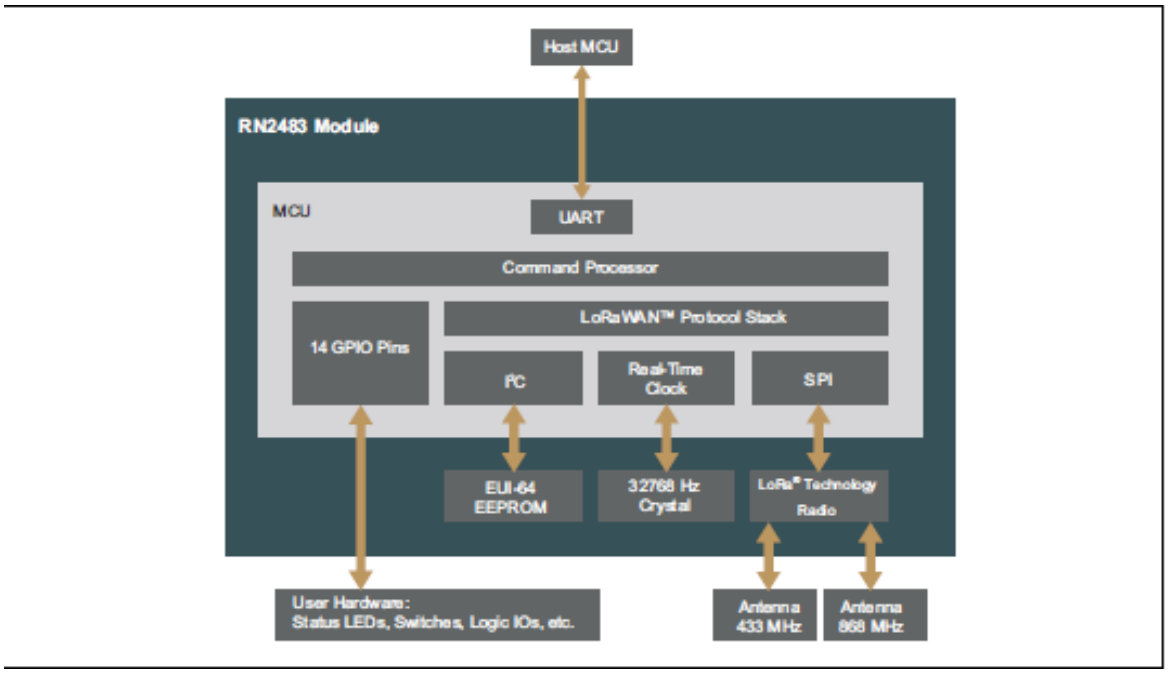

Figura 17: Diagrama de bloques del chip RN2483

El módulo transceptor de LoRa RN 2483 carece de antena incorporada, por lo que debemos llevar en cuenta el diseño de la antena. Es por ello que en el diseño debemos añadir la huella del conector de la antena. Además, debemos tener en cuenta varios requisitos, ya que la implementación de las antenas suele ser delicada. Por lo tanto, antes de diseñar, deberemos tener en cuenta las siguientes características:

- Las antenas requieren de pistas muy anchas, dentro de nuestras posibilidades y de nuestras limitaciones.
- La pista en cuestión, además, debe ser lo más corta posible, y lo más recta posible.
- En nuestro caso particular, puesto que se trata de un shield para Raspberry, debe adecuarse a las medidas de esta. Por ello, el conector debe estar ubicado en uno de los extremos de nuestra placa.

4.1.7 Conector antena SMA Amphenol RF

El conector de antena SMA que vamos a utilizar ofrece un muy buen rendimiento siempre y cuando trabajemos con un rango de frecuencias entre 0 y 26.5GHz. Se trata de un conector que va conectado por ambas caras de la PCB, es decir, dispone de 4 puntos que van conectados a masa, y un punto de conexión con el chip RN 2483. Además, se trata de un conector con una impedancia de 50 Ω.

Podemos apreciar en la imagen como es el componente:

Figura 18: Conector de antena SMA

4.1.8 Descripción módulo XBP24B

El módulo XBP24B es uno de los múltiples chips que podemos encontrar en el mercado, cuyo protocolo de comunicación, como su nombre indica, es el protocolo ZigBee.

El módulo de radiofrecuencia Xbee 802.15.4 es ideal para aplicaciones que requieren baja latencia y tiempos de comunicación predecibles, proporcionando una comunicación rápida y robusta en conexiones punto a punto y multipunto/estrella.

Sus características principales son:

- Banda de frecuencia de 2,4 GHz.
- Velocidad de transmisión de datos de hasta 250 Kbps.
- Alcance aproximado de hasta 120 metros en línea de visión, en el caso de interiores, aproximadamente 40 metros.
- 20 pines Input/Ouput digitales.
- Rango de alimentación entre 2,1V y 3,6V.
- Rango de temperatura entre -40ºC y 85ºC

A continuación podemos apreciar su diagrama de pines, y sus dimensiones:

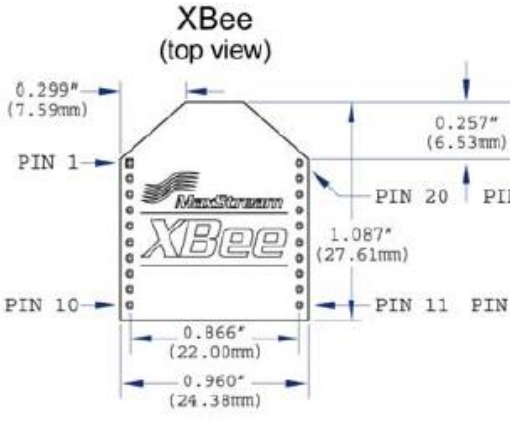

Figura 19: Diagrama de pines del chip XBP24B

4.1.9 NRF24L01

El módulo NRF24L01 es un chip de comunicación inalámbrica fabricado por Nordic Semiconductor. El NRF24L01 integra un transceptor RF (transmisor + receptor) a una frecuencia entre 2.4GHz a 2.5GHz, una banda libre de uso gratuito, y pudiendo elegir entre 125 canales espaciados a razón de 1MHz. La tensión de alimentación del NRF24L01 va desde 1.9 a 3.6V. La velocidad de transmisión es configurable entre 250 Kbps, 1Mbps, y 2 Mbps y permite la conexión simultánea con hasta 6 dispositivos.

El NRF24L01 también incorpora la lógica necesaria para que la comunicación sea robusta, incorporando mecanismos de corrección de errores y reenvío de datos si es necesario, liberando de esta tarea al procesador. El control del módulo se realiza a través del bus SPI, por lo que es sencillo controlarlo desde la Raspberry.

Seguidamente, podemos apreciar el módulo en cuestión:

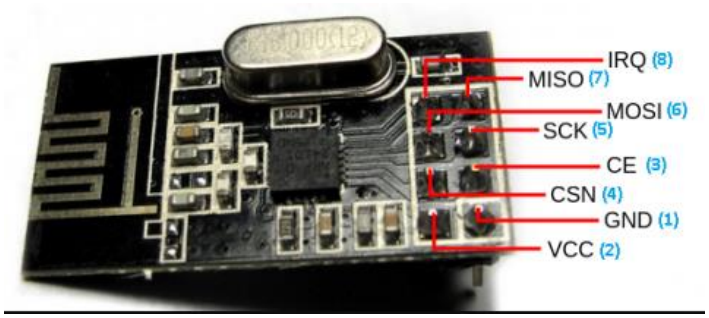

Figura 20: Conexiones módulo NRF24L01

4.2 Diseño PCB con KICAD

Como ya habíamos comentado previamente, la herramienta de diseño que vamos a emplear en la creación de nuestra PCB es KICAD.

En primer lugar, y después de realizar una formación acerca del entorno de la herramienta, de sus componentes, y como funciona, procedemos al diseño del esquemático.

Lo primero que debemos hacer, es la realización del esquema, en el que vamos a diseñar el prototipo de la PCB, como vamos a colocar los componentes, y sus correspondientes conexiones.

KICAD dispone de una librería de componentes bastante limitada, por lo que la gran mayoría de los componentes que vamos a utilizar no existen, es decir, no existen las librerías con los símbolos que vamos a utilizar. Concretamente, de todo lo que necesitamos, solo dispone del símbolo que corresponde al cabezal de la Raspberry PI y el conector del módulo Nordic NRF24L01.

Para ello, utilizaremos el editor de librerías. Esta herramienta nos permite “crear” nuestros componentes.

Vamos a listar los componentes que tenemos que crear:

- Módulo RN 2483.
- Módulo Xbee24B.
- Convertidor lógico (x2).
- 74LVC1G3157GV
- Conector SMA

4.2.1 Diseño de los componentes en KICAD

A continuación, vamos a mostrar uno a uno todos los símbolos creados, en el orden que acabamos de mencionar. Además, también mostraremos el resto de símbolos de los componentes que ya tenemos.

RN 2483

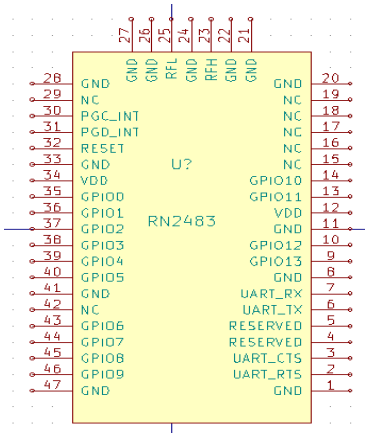

Figura 21: Esquema módulo RN2483

XBP24B

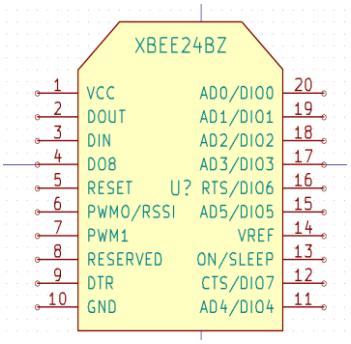

Figura 22: Esquema módulo XBP24B

Convertidor de nivel lógico

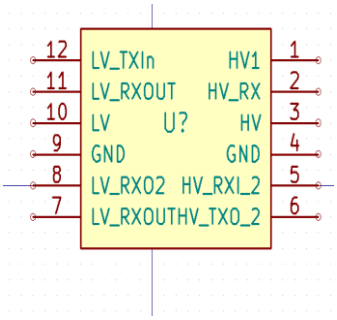

Figura 23: Esquema convertidor de nivel lógico

74LVC1G3157GV

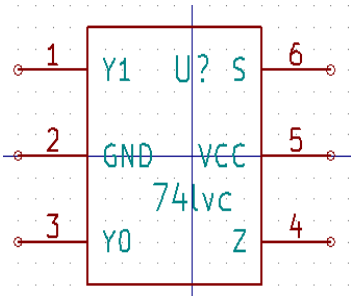

Figura 24: Esquema Mux/Demux

Raspberry Pin Header

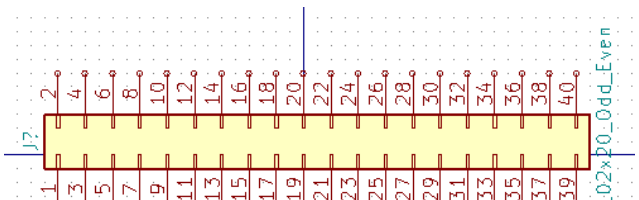

Figura 25: Cabezal Raspberry Pi

NFR24L01

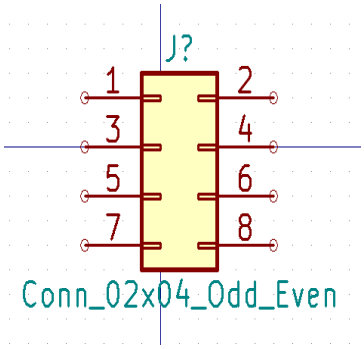

Figura 26: NFR24L01

Conector SMA

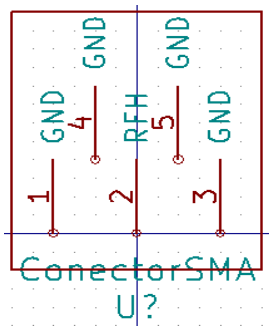

Figura 27: Conector SMA

4.2.2 Diseño esquemático

Una vez tenemos todos los componentes que vamos a necesitar, añadimos todas las librerías creadas a nuestro proyecto, el cual hemos llamado PCB_RASPBERRY:

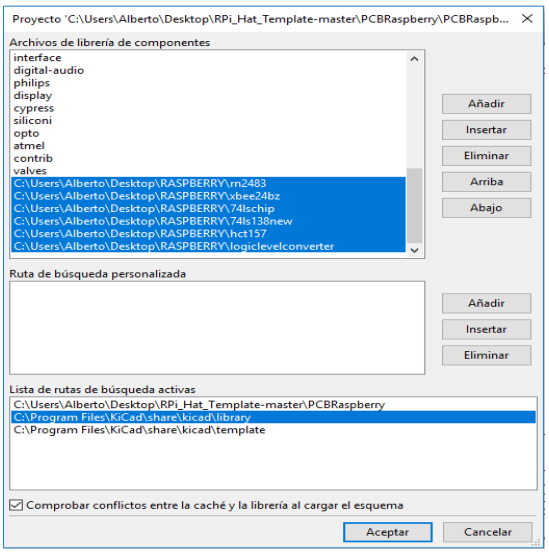

Figura 28: Librerías proyecto

Una vez añadidas las librerías con las que vamos a trabajar, procedemos a diseñar el esquemático, y es aquí donde debemos tener en cuenta todos los requisitos de nuestro sistema, que hemos ido comentando paso a paso previamente. Cabe mencionar que el proceso hasta llegar a obtener el esquemático final ha sido bastante complejo, ya que desde que se empieza a diseñar, hasta que se termina, hay una cantidad de cambios muy importante.

El resultado final del diseño del esquemático podemos apreciarlo en la siguiente página, ya que se trata de una imagen con un tamaño considerable.

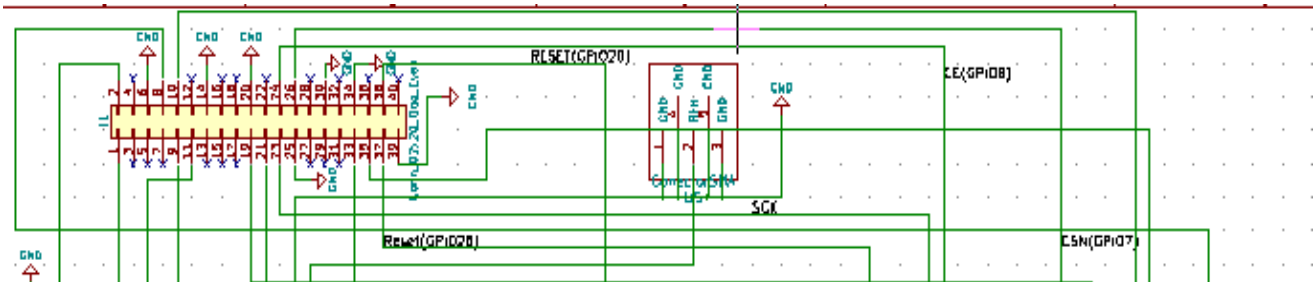

Figura 29: Esquema PCB

Las franjas coloreadas en verde son todas las conexiones que se deben realizar, y las cruces hacen referencia a los pines que no vamos a utilizar, y que por tanto los marcamos como “no conectado”. Seguidamente, después de realizar todas las conexiones, deberemos renombrar todos los componentes, para, posteriormente, realizar el control de reglas eléctricas, diseño de las huellas, asignación de las huellas, y finalmente, generar la NETLIST de nuestro esquemático.

Una vez realizado el esquemático debemos cumplir con una serie de leyes eléctricas. Se trata de un test que incorpora la misma herramienta para comprobar que todo está bien conectado:

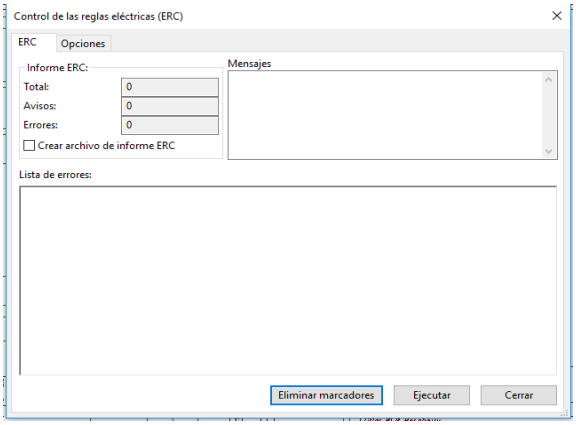

Figura 30: Control de reglas eléctricas

Como podemos comprobar, nuestro esquemático cumple con todas las reglas eléctricas.

4.2.3 Creación de las huellas

Una vez tenemos nuestro esquemático finalizado, y tenemos la certeza de saber que está bien, el siguiente paso es “plasmarlo”, para que un fabricante nos diseñe nuestra placa tal y como queremos. Pero para ello necesitamos la creación de las huellas de todos los componentes que hemos utilizado. En el caso de la edición de librerías de KICAD hemos hecho un diseño bastante preciso, pero no aproximado, es decir, en el diseño de los componentes no hemos tenido en cuenta las medidas exactas del componente.

Y es precisamente lo que ahora debemos hacer. La huella de cada componente quedará grabada en la placa, y es justo en estas huellas dónde, posteriormente, soldaremos todos nuestros componentes.

Por lo tanto, debemos coger todos los datasheet de los componentes, ya que es ahí donde encontraremos los apartados donde el fabricante especifica las medidas exactas para luego poder soldar.

Podemos apreciar que cada componente está compuesto por un determinado número de PADS, y que para poder realizar las diferentes conexiones entre los componentes, será necesario el uso de vías que permitan conectar las dos caras de nuestra placa de circuito impreso.

4.2.3.1 ¿Qué es un pad?

Un **PAD** es una superficie de cobre en un circuito impreso o PCB que permite soldar o fijar el componente a la placa.

Existen dos tipos de pads:

- **Thru-hole (Pasante)**: Son todos aquellos componentes que poseen pines para ser instalados en perforaciones metalizadas (llamadas thru-hole pads). Este tipo de componentes se suelda por la capa opuesta. Generalmente son montados por un solo lado de la placa.
- **SMD (montaje de superficie)**: Son todos aquellos componentes que se montan de forma superficial. Tienen la ventaja de que pueden montarse por ambos lados, además de ser más pequeños que los thru-hole, lo que permite hacer circuitos más pequeños y densos. Son interesantes para diseños en alta frecuencia debido a su tamaño reducido.

4.2.3.2 ¿Qué es una vía?

Una vía es una perforación metalizada que permite que la conducción eléctrica no se interrumpa cuando se pasa de una superficie a otra, es decir, las vías se utilizan para conectar los componentes que se encuentran en diferentes caras en la PCB.

4.2.4 Editor de huellas

Ahora que nos hacemos una pequeña idea de que partes está compuesta una huella de un componente, vamos a comentar la nomenclatura que vamos a utilizar.

A diferencia del editor de huellas de KICAD, que dispone de escasas librerías, KICAD ofrece una extensa librería de componentes, que podemos descargar incluso del repositorio de GITHUB [28], por lo que en este aspecto dispone de unas excelentes librerías. Pero como no, las huellas de casi todos nuestros componentes hemos tenido que hacerlas a mano. Únicamente, y de forma similar, hemos podido aprovechar las huellas de 2 componentes, y es que se trata de dos cabezales:

- **PIN HEADER 2x20- Pitch** [29] **2.54mm:** Se trata del componente que permite conectar nuestra PCB con la Raspberry.
- **PIN HEADER 2x04 – Pitch 2.54mm:** Se trata del componente que permite conectar el módulo NRF24L01 a la PCB.

Para el diseño de las huellas, utilizaremos el módulo PCB New, de Kicad, que nos permite, como acabamos de decir, diseñar las huellas de los componentes elegidos.

A continuación, vamos a poder ver, una a una, todas las huellas de los componentes de nuestra placa de circuito impreso.

Cabezal Raspberry Pi

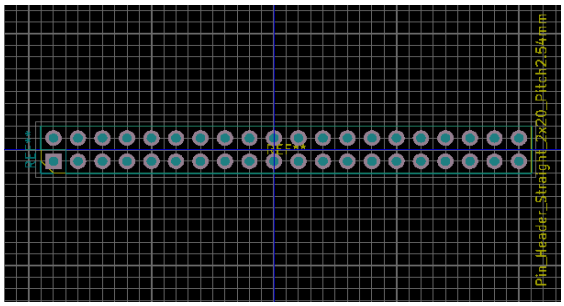

Figura 31: Huella Cabezal Raspberry Pi

Tipo de PAD: PASANTE

Forma: CIRCULAR

Tamaño X: 1,7mm

Tamaño taladro: 1mm

Cabezal NRF24L01

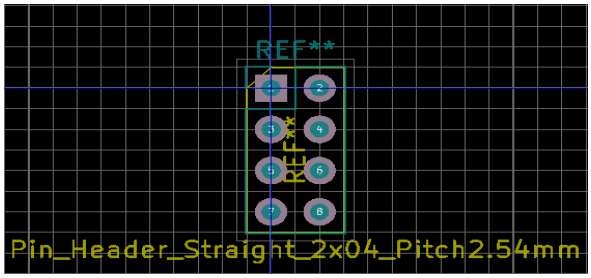

Figura 32: Huella Cabezal NRF24L01

Tipo de PAD: PASANTE

Forma: CIRCULAR

Tamaño X: 1,7mm

Tamaño taladro: 1mm

74LVC1G3157GV

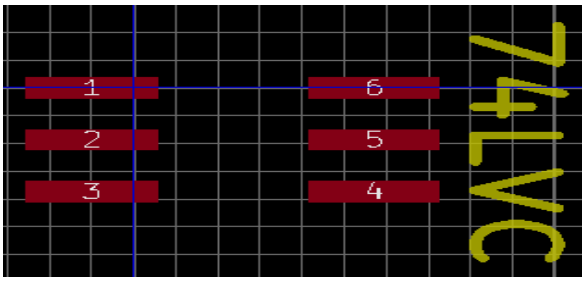

Figura 33: Huella 74LVC1G3157GV

Tipo de PAD: SMD

Forma: Rectangular

Tamaño X: 0,4mm

Tamaño Y: 1,6mm

RN 2483

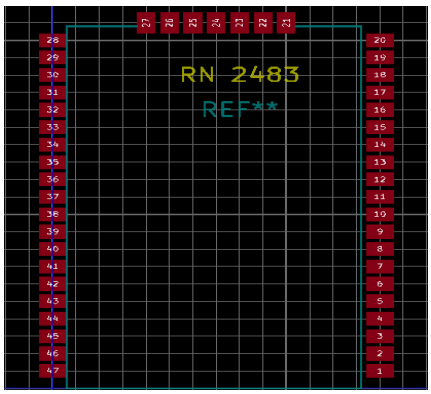

Figura 34: Huella RN 2483

Tipo de PAD: SMD

Forma: RECTANGULAR

Tamaño X: 1.524 mm

Tamaño Y: 1.016 mm

XBP24BZ

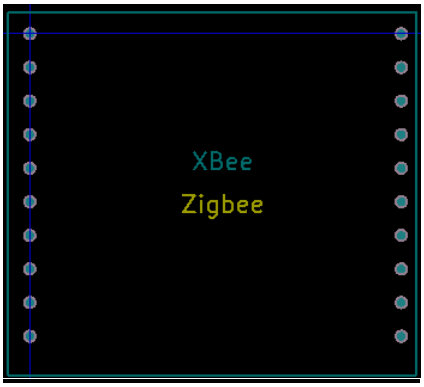

Figura 35: Huella XBP24B

Tipo de PAD: PASANTE

Forma: Circular

Tamaño X: 1,3 mm

Tamaño Taladro: 0,7

Convertidor de nivel lógico

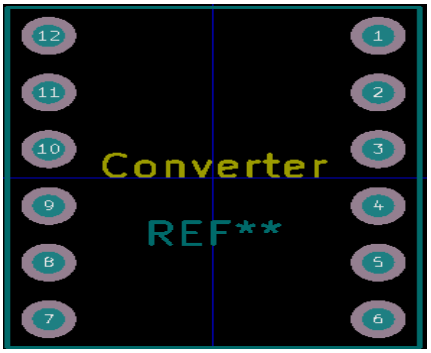

Figura 36: Huella convertidor

Tipo de PAD: PASANTE

Forma: Circular

Tamaño X: 1.7 mm

Tamaño taladro: 1mm

Conector SMA

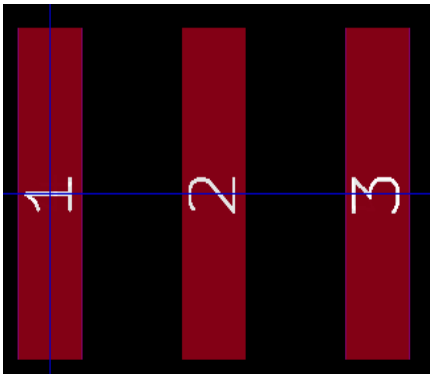

Figura 37: Huella conector SMA

Tipo de PAD: SMD

Forma: RECTANGULAR

Tamaño X: 1.02mm

Tamaño Y: 4,5mm

Las medidas de cada pad vienen definidas en el *datasheet* de cada componente. Es ahí donde nos indica:

- Forma del PAD.

- Tipo de PAD.
- Medida del PAD.
- Distancia entre PADS.

Es el momento, pues, de asignar cada una de las huellas que hemos creado a su componente, ya que debemos asociar a ese componente su huella correspondiente, y seguidamente generaremos la NetList.

La NetList nos crea un archivo donde se asocian los componentes con sus huellas, y estas con las conexiones de cada componente. Para asignar a un componente su huella, debemos editar el componente:

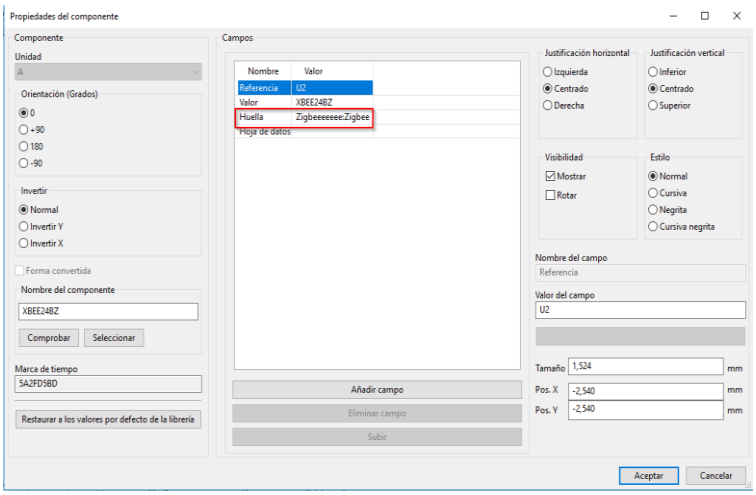

Figura 38: Asignación de huellas

Debemos hacer este proceso en todos y cada uno de los componentes, para asignarle a cada uno su huella.

Una vez asignadas todas las huellas de los componentes, procedemos, a asignar a cada componente su referencia:

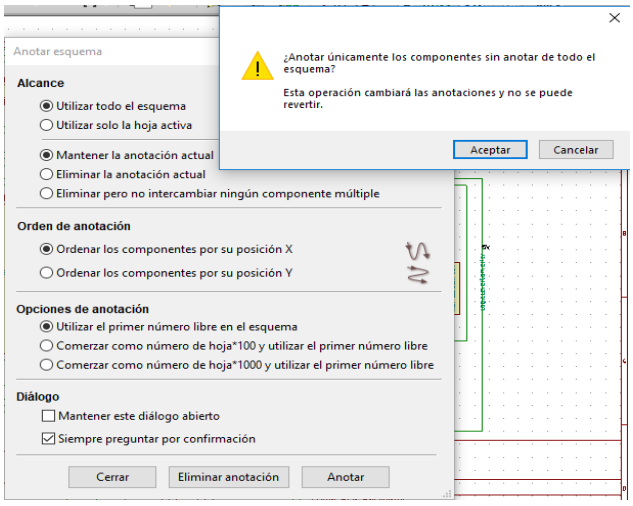

Figura 39: Asignación referencia a cada componente

Y por último, antes de proceder al diseño de la placa, generaremos nuestra NetList:

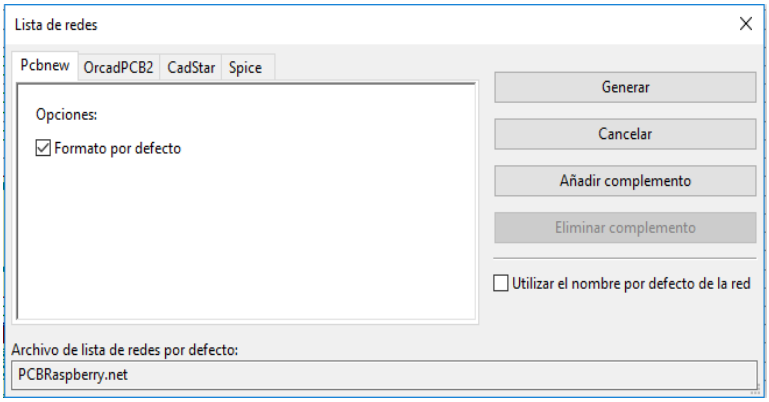

Figura 40: Generación NetList

4.2.5 PCBNEW- Editor de placas de circuito impreso

Y llegamos al tramo final en el diseño de la placa de circuito impreso. Después de todo el proceso aquí mencionado llegamos a la parte clave del diseño. Es aquí donde el resultado de todo el proceso puede acabar bien, o puede acabar mal. Es aquí donde, después de distribuir nuestras conexiones, vamos a determinar las tolerancias del sistema y donde vamos a determinar las reglas de diseño, quizá una de las cosas más importantes en el diseño de una placa.

Pero vamos a paso, en primer lugar definiremos nuestras reglas de diseño, posteriormente leeremos la *NetList* que hemos creado, y a partir de esta combinación, podremos empezar a distribuir el diseño de la placa.

4.2.5.1 Reglas de diseño

En este apartado es donde podemos editar las reglas de diseño, en función de los requerimientos de nuestro sistema.

Por lo tanto, en primer lugar, vamos a partir de las reglas habituales de una PCB con ancho de pista de cobre de 35 micras. Se trata de una medida muy corriente en el diseño de una PCB.

El ancho de la pista se determina en función de la corriente máxima que vaya a soportar la pista. Puesto que se trata de señales digitales, es muy poco probable que se superen corrientes superiores a 500mA.

Según las reglas de diseño existentes, tenemos una relación entre el ancho de pista, y la corriente máxima. Puesto que los pines de la Raspberry Pi tienen un pico de corriente máximo de 16mA, a partir de la siguiente tabla podemos determinar cuál será el ancho de pista que vamos a utilizar:

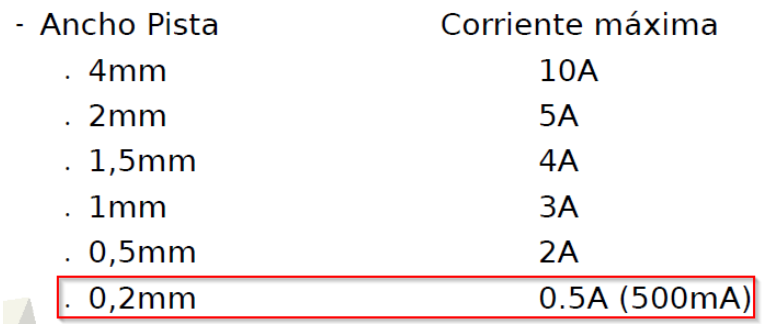

Figura 41: Ancho de pista – Corriente máxima

Por lo tanto, llevando en cuenta todo lo aquí mencionado, utilizaremos pistas con un ancho de 0.25mm, más que suficiente para la corriente máxima que puede generar la Raspberry PI.

Como vamos a utilizar pistas de 0,25mm de ancho, la separación entre pistas, por ende, va a ser igual o mayor que 0,25mm. En nuestro caso, vamos a definir un margen de 0,25 mm.

Debemos rellenar, además, otras opciones que nos proporciona la herramienta. Hablamos de los siguientes parámetros:

- Diámetro Vía.
- Taladro Vía.
- Diámetro Microvía.
- Taladro Microvía.

KICAD permite desactivar la opción para las microvías, de manera que, como nosotros no vamos a utilizar ninguna microvía, dejaremos constancia de “No permitir microvías”.

Además, vamos a utilizar vías con un diámetro de 0,60 mm.

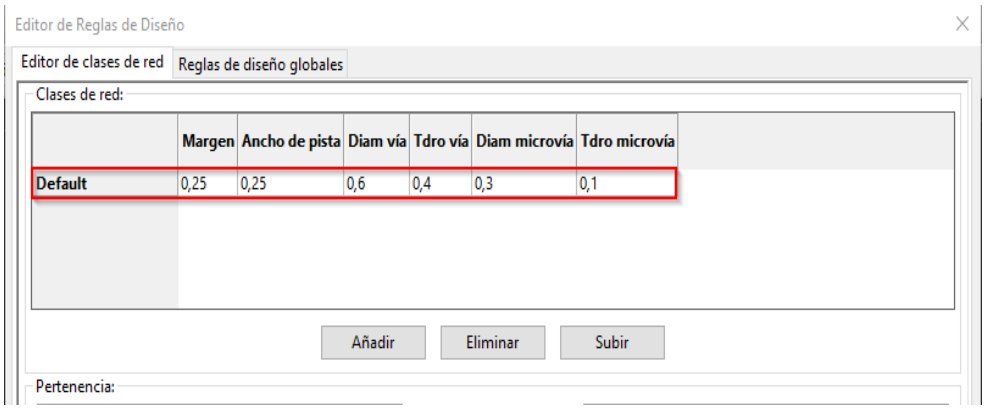

Figura 42: Reglas de diseño de nuestro esquemático

Y llegados a este punto, en el que hemos definido nuestras reglas, ya podemos leer nuestra NetList:

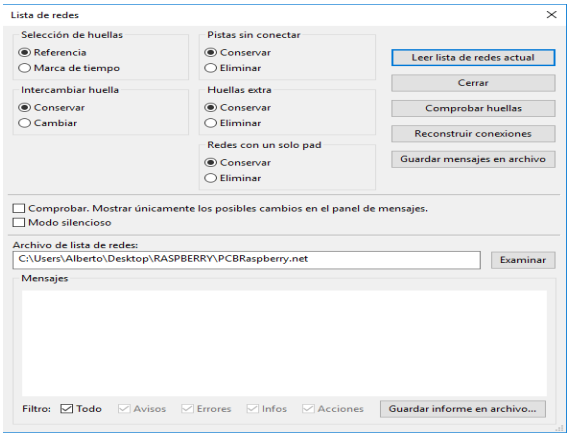

Figura 43: Lector de NetList

Leemos nuestra lista, llamada “PCBRaspberry.net”, y si todo está correcto, debe compilar el archivo y cargarlo en *PCBNew.*

Si no tenemos ninguna contraorden, ya podemos empezar, pues, con la distribución de las huellas de los componentes, y con la distribución de las conexiones de cada componente, que, como ya se ha mencionado, será una shield con doble cara.

Debemos tener nuevamente una serie de consideraciones a la hora de distribuir y colocar nuestros componentes:

- No pueden haber pistas que se crucen entre sí sobre la misma cara.
- Debemos procurar que las pistas pasen a una distancia razonable de los pads de los componentes, ya que una ubicación muy próxima puede dificultar el proceso de soldadura de los componentes.
- No utilizaremos microvías ya que la placa no las requiere.
- Es muy importante tener una ubicación óptima para el componente RN 2483, ya que debemos colocar el conector de la antena de este dispositivo en un extremo de la placa.
- Debemos evitar, si es posible, ángulos de 90º a la hora de trazar las pistas.
- Evitar ángulos agudos.

A pesar de tener los conocimientos teóricos acerca de cómo se diseña una placa, quiero recalcar la cantidad de veces que se debe realizar un diseño hasta ponerlo en práctica.

4.2.5.2 Consideraciones hasta llegar al diseño final

En este apartado quiero hacer especial hincapié en la dificultad que tiene diseñar una placa que cumpla con todas las especificaciones de todos los componentes. Con ello me refiero, en primer lugar, a la colocación de los chips. Debido a todos los diferentes componentes que tendrá la placa, y cada uno de ellos con unas determinadas dimensiones físicas, se hace un proceso muy repetitivo el cómo poder colocar cada uno de ellos de manera que no se toquen entre sí. Esto no supondría ningún problema si las medidas de la shield no

fueran un problema. Pero no es así. Pese a que no necesitamos una placa excesivamente grande, la Raspberry tiene unas medidas determinadas, y en cualquier caso, no se aconseja que la shield que se quiera incorporar sea mucho más grande que esta, por lo que tenemos restricciones en lo que a medidas se refiere.

Figura 44: Raspberry Pi 3 model B+

Concretamente, nuestro modelo de Raspberry tiene unas medidas de 85 x 56 mm.

Nuestra shield tendrá unas medidas muy similares, intentando cumplir con los requerimientos del sistema.

Además, como también se puede comprobar, la Raspberry tiene todos los componentes a la vista, por lo que también, debemos tener cuidado para que nuestros componentes no toquen con los componentes de la Raspberry. Normalmente los problemas vienen derivados con los conectores USB que incorpora la placa.

Nuevamente, llevando en cuenta todas estas consideraciones, y reiterando una y otra vez el proceso de distribución y colocación tanto de componentes como de pistas, podemos mostrar el resultado final, que sabemos con certeza, antes de mandarlo a fabricar, que se trata de un modelo que encaja perfectamente con la Raspberry Pi. Para ello, KICAD dispone de una herramienta que nos permite hacer una impresión previa sobre papel, a tamaño real, de nuestra placa. Puesto que se trata de una impresión bastante precisa, podemos recortar el esbozo de nuestra placa, y podemos saber en qué punto estamos, y cuánto debemos mover los componentes para poder cumplir con las medidas que se exigen. Cabe mencionar que, en prácticamente todas las ocasiones, el hecho de tener que mover componentes, implica tener que moverlo todo de nuevo, incluidas las conexiones, por lo que quiero realzar el arduo trabajo que tiene hasta poder llegar a lo que se quiere.

Finalmente, el resultado es el siguiente:

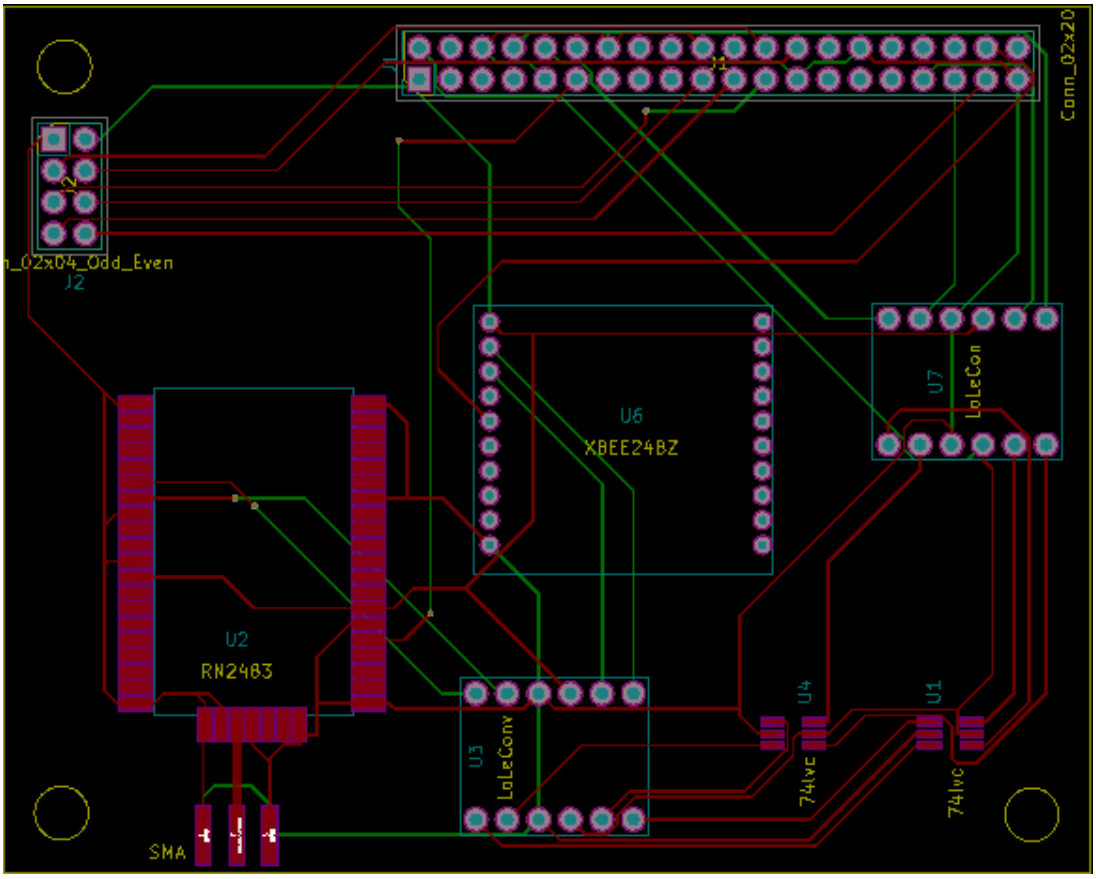

Figura 45: Diseño final circuito impreso

4.2.5.3 Creación de un plano de tierra

El siguiente paso es crear un plano de tierra para cada una de las dos caras de nuestra placa, pero, ¿qué es un plano de tierra?

En esencia, un plano de tierra consiste en tomar todo el espacio no utilizado en una placa de circuito impreso, y conectarlo a la red de tierra. Con el plano de tierra conseguimos evitar la inducción de corrientes parásitas, por lo que se trata de una opción muy recomendable en circuitos de doble capa.

Pues bien, aquí vamos a mostrar el resultado del plano de masas que hemos creado, tanto para la cara superior, como para la cara inferior:

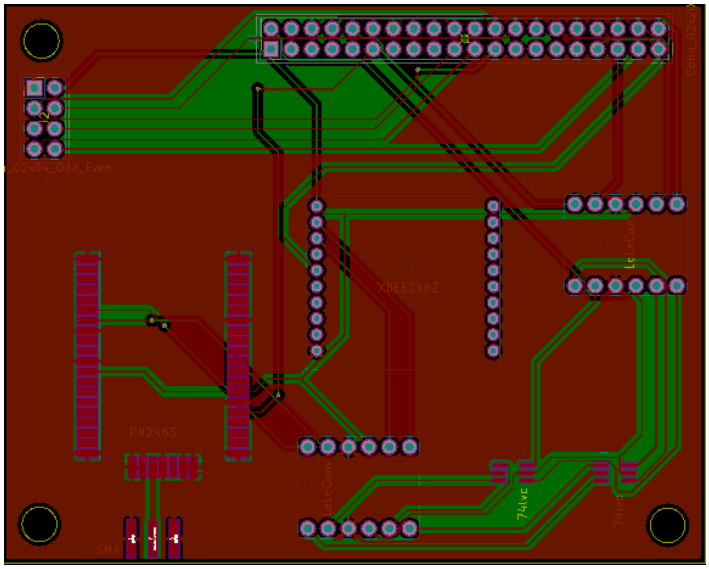

Figura 46: Plano de tierra capa superior

Y, seguidamente, podemos ver el plano de masa para la cara posterior:

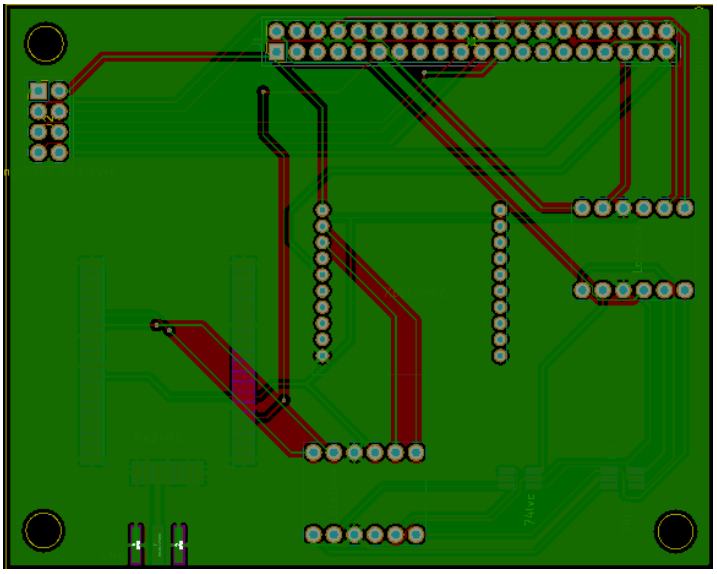

Figura 47: Plano de tierra capa opuesta

4.2.6 Fabricación de la placa

En este punto, llegamos al momento de mandar a fabricar nuestro prototipo, para convertirse en una realidad.

Para ello, vamos a enviar al fabricante los archivos que se requieren para fabricar una placa. Éstos son los llamados archivos Gerber. Los archivos Gerber contienen la información necesaria para la fabricación de un circuito impreso. Se trata de archivos ASCII con coordenadas e instrucciones simples que permiten interpretar el circuito impreso a fabricar independientemente del sistema de diseño utilizado.

Los archivos Gerber que vamos a generar, y que el fabricante requiere para poder fabricar la placa son los siguientes:

- **B.cu.gbr** → Capa inferior de cobre.

- **B.SilkS.gbr** → Serigrafía inferior.
- **Edge.Cuts.gbr** → Esquema del tablero. Esto hace mención a la línea de corte que se utiliza para delimitar las dimensiones de la placa.
- **F.Cu.gbr** → Capa superior de cobre.
- **F.SilkS.gbr** → Serigrafía superior.
- **NPTH.drl** → Esquema de taladros.
- **PTH.drl** → Esquema de taladros.

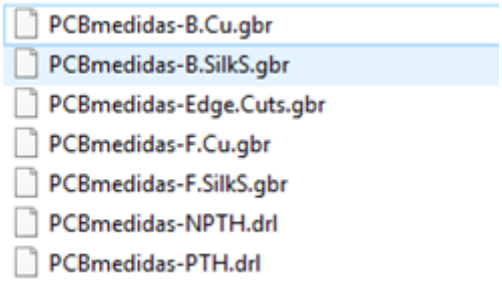

Figura 48: Archivos Gerber generados

A partir de aquí, ya podemos elegir al fabricante que queramos.

En este caso, y puesto que es un proyecto que está vinculado a la empresa AEInnova, decidimos que las placas las fabricará el fabricante PCBWay [30], ya que han trabajado con ellos en reiteradas ocasiones, y siempre con un resultado satisfactorio.

El pedido de la placa se realiza a través del portal web del fabricante, y es aquí donde se especifica las características que queremos que tenga nuestra placa y la cantidad de estas que se quieren adquirir. Normalmente, un fabricante de placas de circuito impreso no suele fabricar una sola placa, por lo que las tarifas varían por el determinado volumen que se requiera.

Podemos ver un patrón de la plantilla que ofrece el fabricante cuando se quiere realizar un pedido a continuación:

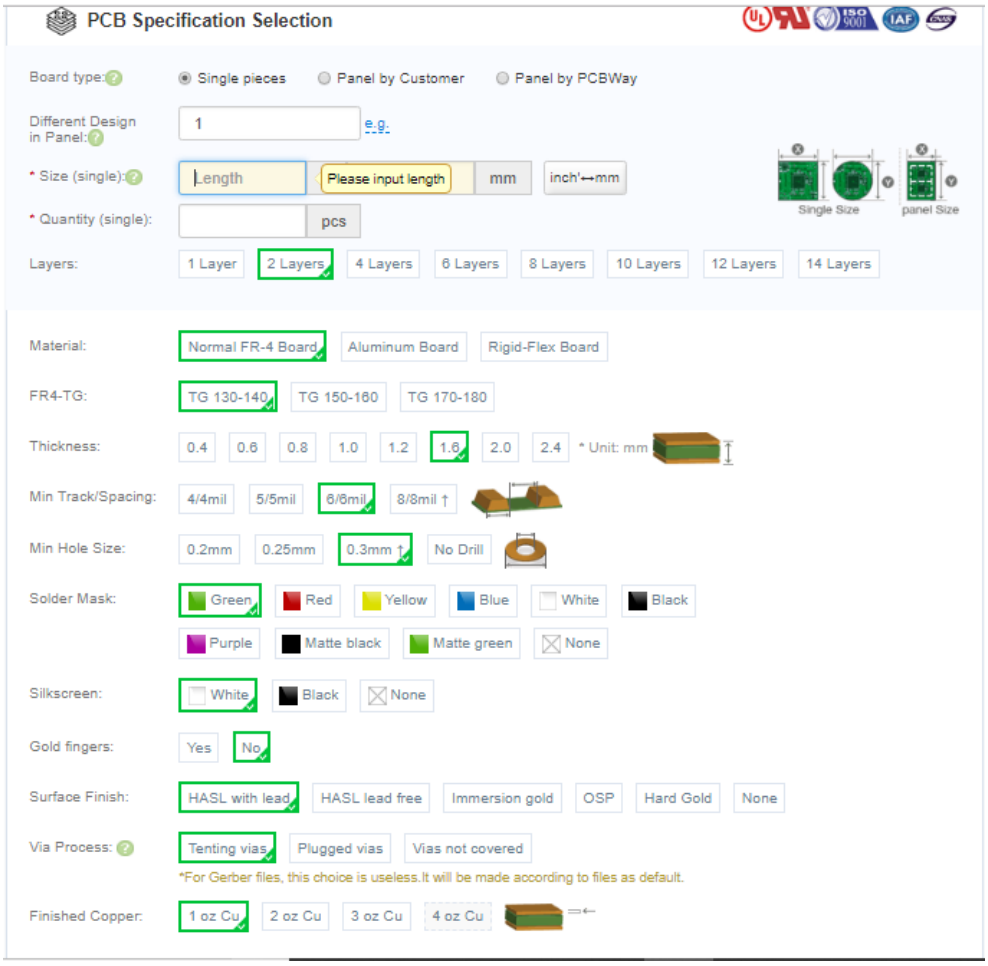

Figura 49: Plantilla para realizar el pedido de una placa de circuito impreso

Finalmente, debemos esperar un tiempo aproximado de unos 8-10 días para recibir la placa.

4.2.7 Soldadura de los componentes de la placa

Uno de los últimos escalones para poder finalizar satisfactoriamente nuestra placa de circuito impreso es el proceso de soldadura de los componentes a la placa. Se trata de uno de los procesos más delicados, ya que utilizamos componentes muy pequeños, y no disponemos de herramientas de soldadura profesional, o de máquinas que realicen esta tarea, por lo que utilizaremos un soldador de estaño convencional.

Para llegar a obtener un resultado óptimo, recibimos la ayuda por parte de miembros de la universidad con una larga trayectoria profesional en el mundo de la electrónica, y por ende, con una experiencia más que contrastada para realizar soldaduras.

Tenemos una mayor dificultad a la hora de soldar los componentes cuyos pads son de superficie, ya que éstos tienen unas dimensiones muy reducidas, y es muy fácil equivocarse.

El resultado final de todo este proceso podemos apreciarlo en la siguiente imagen:

Figura 50: Resultado final después de soldar los componentes

Finalmente, y después de meses de trabajo, tenemos conformada nuestra shield Wireless multiprotocolo.

Pero este no es el final, ya que, pese a que el trabajo ya está plasmado, aún queda un largo camino por recorrer. En primer lugar, debemos hablar acerca de las posibilidades que tiene la placa, de las líneas futuras de las que se va a derivar, planes a corto plazo, y como no, antes de todo esto, debemos asegurarnos que la placa funciona correctamente.

Para ello, vamos a realizar una serie de test que nos permitan asegurar que funciona.

4.2.8 Test de comprobación

La prueba definitiva para demostrar que todo nuestro trabajo ha llegado a buen puerto es poder demostrar que la placa funciona correctamente. Para ello, vamos a demostrarlo mediante la utilización de uno de los módulos insertados en la placa. Vamos a trabajar con el controlador del protocolo ZigBee, ya que dispone de herramientas muy visuales y sencillas de utilizar para realizar una comunicación entre dos módulos que trabajen con este protocolo.

4.2.8.1 Test de comprobación módulo XBP24BZ

Para ello, vamos a necesitar, además de nuestra shield, los siguientes componentes:

- 2 módulos XBP24BWIT
- 1 arduino
- 1 arduino Shield
- 1 Raspberry Pi

- 1 shield para Raspberry Pi

Para ponernos en situación, vamos a ver cómo funciona exactamente una red Zigbee.

En este caso, estamos interesados en la topología de malla (mesh), explicada anteriormente. Queremos que nuestros módulos se comuniquen directamente entre ellos, sin tener que utilizar un módulo que haga de nodo entre ellos.

Para ello debemos configurar los módulos XBP24WIT debidamente, ya que queremos que uno funcione como coordinador de red, y otro funcione como dispositivo final, para poder ver como se comunican, y para comprobar que nuestra shield funciona correctamente. Podemos ver a continuación la topología de la que estamos hablando:

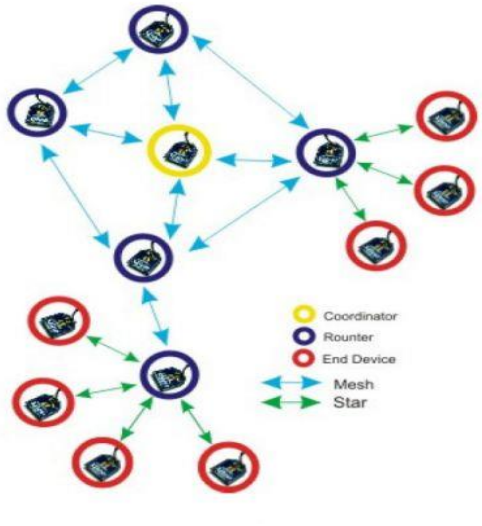

Figura 51: Red mesh en protocolo ZigBee

En nuestro caso no disponemos de dispositivos finales, ya que no es el cometido de estas pruebas. Únicamente queremos comprobar que la shield funciona.

Para configurar los dispositivos en cuestión, haremos uso de una herramienta muy útil, llamada XCTU [31].

XCTU es un software multiplataforma que permite interactuar con los módulos Xbee mediante una interfaz gráfica. Utilizamos esta aplicación ya que incluye herramientas que hacen muy sencillo configurar y probar los módulos Xbee.

Una de las ventajas de XCTU es que nos sirve para configurar, inicializar, actualizar firmware y testear los módulos XBee, comunicándose por puerto serie a los módulos.

El primer paso de todos es configurar los módulos para que trabajen con la función que queremos. Como ya hemos dicho, necesitamos que uno de los módulos actúe como un dispositivo final, y el otro de los módulos actuará como coordinador. En las siguientes imágenes podemos ver su configuración:

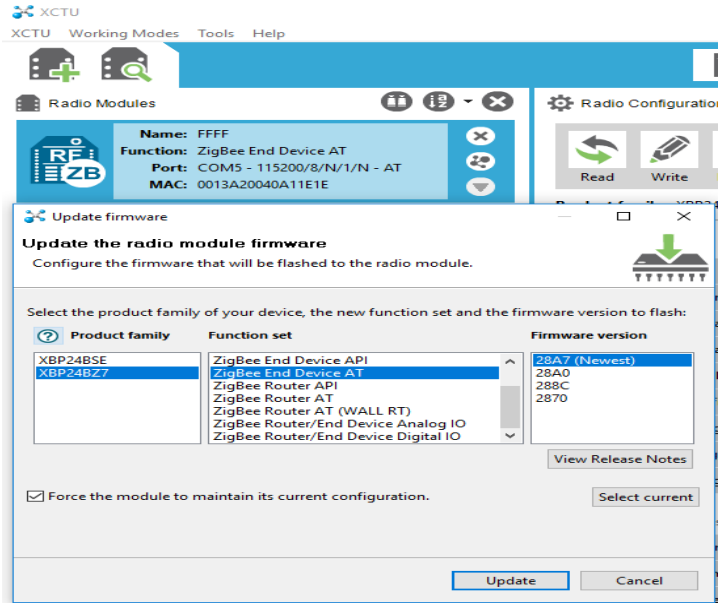

Figura 52: Configuración como dispositivo final

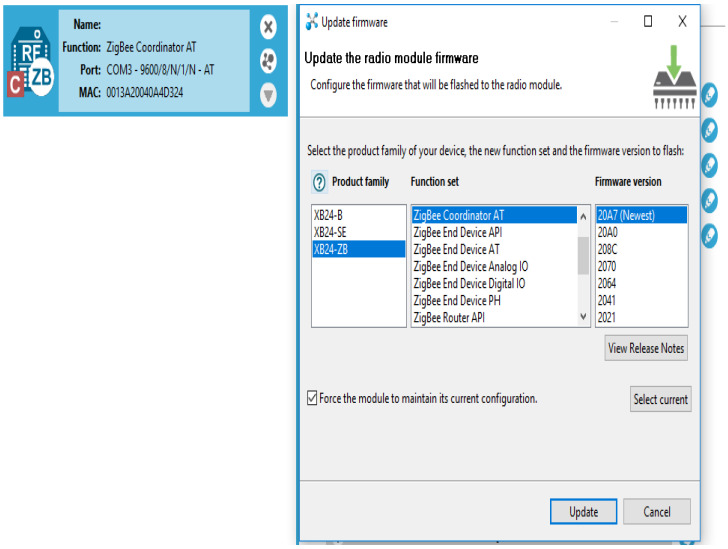

Figura 53: Configuración como coordinador

El siguiente paso, una vez finalizada la configuración de los módulos es hacer que se puedan comunicar entre sí. En este caso, necesitamos cruzar sus direcciones, y establecer que compartan un mismo canal. Como podemos ver a continuación, modificaremos los siguientes campos:

- **PAN ID**: 1234 → Tendrá el mismo valor para ambos módulos.
- **Scan Channels** : 7FFF → Tendrá el mismo valor para ambos módulos,
- **Destination Address High** → En el caso del coordinador, tendrá el valor 13A200, que sería el “Serial Number High” del módulo que actúa como dispositivo final.
  En el caso del dispositivo final, rellenaremos el campo con el valor de “Serial Number high” del coordinador, que en nuestro caso, en ambos chips vale lo mismo

- **Destination Address Low** → En el caso del coordinador, tendrá el valor 40A11E1E, que sería el valor del campo “Serial Number Low” del módulo que actúa como dispositivo final.
  En el caso del dispositivo final, rellenaremos el campo con el valor del campo “Serial Number Low” del módulo que actúa como coordinador.

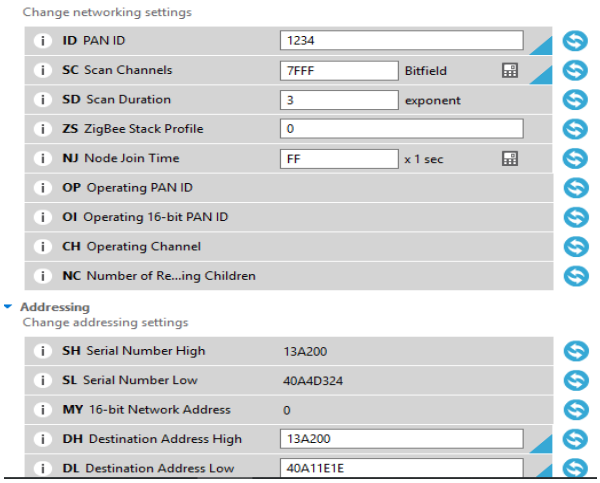

Figura 54: Configuración de los módulos XBee

Una vez configurados ambos módulos, para hacer nuestra prueba, conectamos el módulo coordinador a nuestra Raspberry, a través de la shield que hemos diseñado, para poder demostrar que funciona perfectamente. Aquí podemos apreciar el escenario en cuestión:

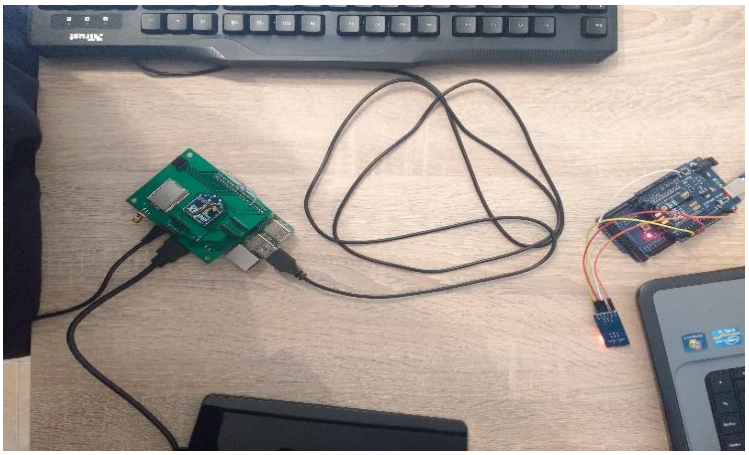

Figura 55: Escenario Raspberry Pi + Shield – Arduino + Arduino shield

En el lado de la Raspberry, por defecto, no podemos acceder a los pines Tx y Rx, por lo que debemos realizar una serie de pasos que permitan acceder a los pines Tx y Rx. Una vez habilitados, debemos la herramienta “minicom”. Esta herramienta nos permite conectarnos por el puerto serie a otro dispositivo, en nuestro caso nos conectaros al módulo que actúa como dispositivo final.

La prueba consiste en enviar un mensaje desde el módulo que actúa como coordinador, y esperar, si todo está bien configurado, a que el mensaje llegue a nuestro dispositivo. De esta forma, podemos comprobar que todo funciona perfectamente, y es un claro indicativo de que la placa funciona perfectamente.

Una vez abierta la herramienta minicom, vamos a enviar el siguiente mensaje, y vamos a esperar que nuestro dispositivo final lo reciba. El mensaje enviado será el siguiente:

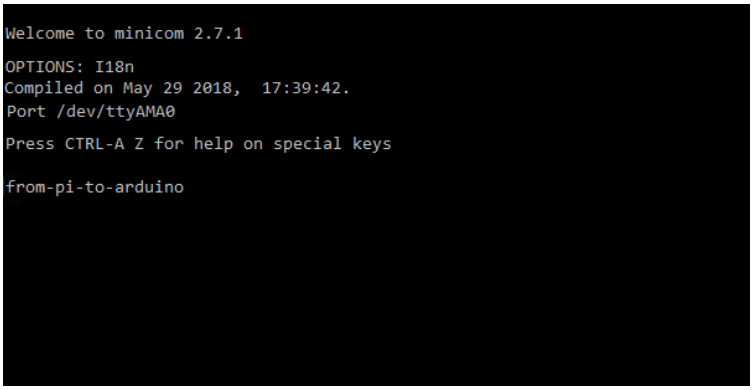

Figura 56: Mensaje enviado desde la Raspberry Pi

Y, como se puede comprobar a continuación, si observamos nuestro módulo que actúa como dispositivo final, obtenemos lo siguiente:

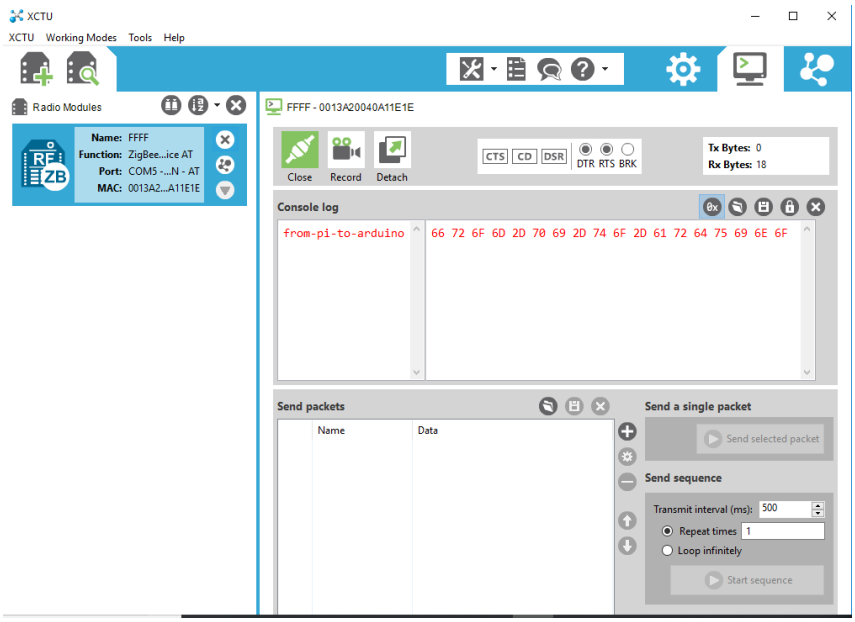

Figura 57: Recepción mensaje enviado a través de la Raspberry PI

Por lo tanto, acabamos de demostrar que nuestra placa, con la debida programación, funciona perfectamente. Hemos conseguido realizar una comunicación entre la Raspberry Pi y un módulo Xbee conectado a un arduino. Esto no es más que una simple demostración de que la placa funciona correctamente, y posteriormente vamos a estudiar las posibilidades de la placa.

4.2.8.2 Comunicación con módulo NRF24L01

En lo que se refiere a la demostración de cómo funcionan los otros dos módulos colocados en la placa, el módulo NRF y el módulo de LoRa, vamos a explicar cómo se debería hacer, y de esta manera dejar unas pautas para quien deba seguir con su desarrollo.

El módulo NRF24L01, como sabemos, se conecta mediante el bus SPI con la Raspberry, así que en primer lugar debemos habilitar la interfaz SPI de la Raspberry, si aún no lo está. Escribiremos en el terminal:

Sudo Raspi-config,

Una vez hayamos entrado en el menú de configuración, buscaremos: Opciones avanzadas → SPI → Habilitar interfaz SPI

Una vez habilitado el bus debemos instalar una de las librerías que nos permite realizar conexiones con el bus SPI. La librería en cuestión es la librería “WiringPI”.

Básicamente la librería que se debe utilizar únicamente dispone de dos funciones, una para la configuración del bus, y otra para leer/escribir:

- **int wiringPiSPISetup (int channel, int speed)**: Esta función sirve para configurar el bus SPI. En la variable llamada “cannel” debemos poner 1 o 0, según el puerto que queramos utilizar, y en la variable “speed” ponemos la velocidad en Herzios que queramos. Si la función nos devuelve un -1 quiere decir que no se ha iniciado bien y que tenemos un error.

- **int wiringPiSPIDataRW (int channel, unsigned char *data, int len)**: Por último, disponemos de la función de lectura/escritura. Esta función está conformada por tres variables. El canal, el buffer de datos y el tamaño de los datos a enviar. Dado que esta función lee y escribe al mismo tiempo, lo que pongamos en la variable “data”, será sobrescrito por lo que se lea en el bus SPI.

El siguiente paso consiste en cargar los drivers SPI en nuestra Raspberry. Para ello ejecutaremos en el terminal la siguiente instrucción:

Gpio load spi

El siguiente paso sería determinar el tamaño del buffer SPI. Por defecto este ya tiene un tamaño de 4KB, así que será más que suficiente para el tipo de transmisión que se pueda emplear. Recordamos que la idea final es poder determinar la calidad de la señal, así que será más que suficiente.

Ahora vamos a recordar algunos detalles del bus SPI de la Raspberry. En primer lugar hay que decir que la Raspberry Pi tiene dos puertos SPI, el puerto 0 y el puerto 1. La velocidad a la que puede comunicarse va desde los 500 KHz hasta los 32 MHz.

Por último, una vez disponemos de la configuración aquí especificada, deberíamos desarrollar la aplicación que se requiera, utilizando librerías para NRF2401, de las cuales hablaremos en el siguiente apartado. Una vez el desarrollo del programa llegue a su fin, deberíamos

conectarnos, al igual que hicimos con el XBee, a un arduino, al que conectaremos el módulo NRF24L01 de la siguiente manera:

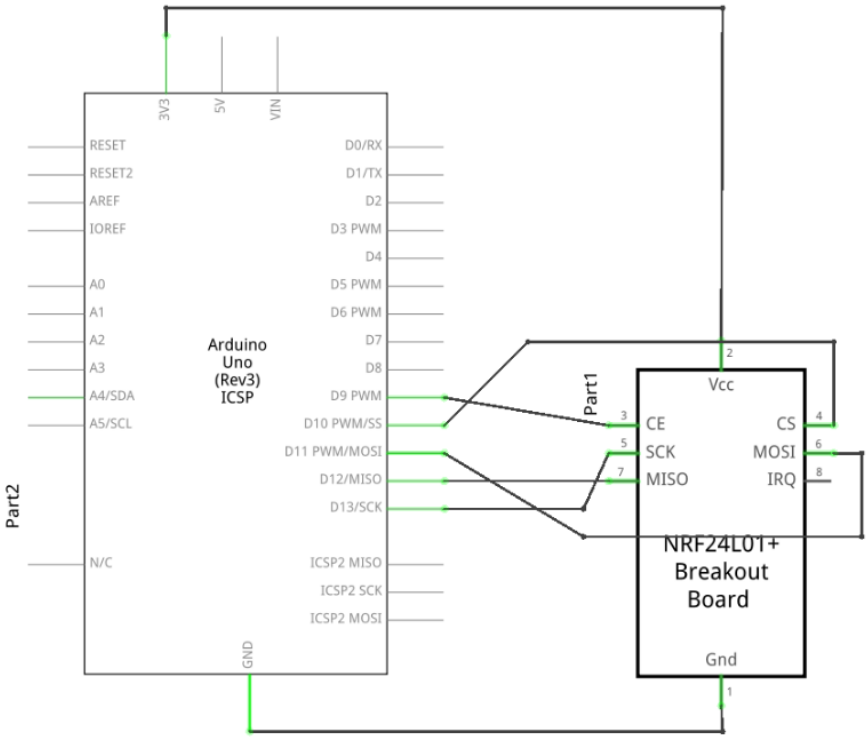

Figura 58: Diagrama de conexión entre módulo NRF24L01 y Arduino

De la misma forma que en la Raspberry Pi, deberíamos programar en la placa Arduino lo que queramos que haga nuestro chip. Como ya hemos mencionado, la parte de desarrollo corresponde a la siguiente fase del proyecto, por lo que aquí se establecen únicamente las bases para conectar los chips y cómo establecer una conexión.

4.2.8.3 Comunicación con módulo RN2483

Finalmente, vamos a detallar como podemos comunicarnos con el módulo RN 2483. Como ya especificamos, nuestro módulo se debe conectar con un Gateway, por lo que en este caso, es difícil demostrar cómo hacer una comunicación, ya que no disponemos de un Gateway, pero aún y así vamos a especificar como se debería hacer.

Como ya sabemos a estas alturas, el modo de comunicación de la Raspberry con el módulo RN2483 se hace mediante el bus UART de la Raspberry. Puesto que únicamente disponía de un puerto UART Tx, y un puerto UART Rx, debimos multiplexar las entradas y salidas de este módulo y del Xbee, para hacer que pudieran funcionar, en el caso que así sea, simultáneamente.

El funcionamiento es el siguiente, el módulo RN2483 debe comunicarse con el Gateway, para que este procese los datos.

Por lo tanto se debe configurar la conexión con el Gateway, en primer lugar, y en segundo lugar, configurar el módulo para poder enviar / recibir datos. Por lo tanto se deberá desarrollar un script para que se registren los datos que nos llegan desde el Gateway, a través del puerto serie, y almacenarlos en la Raspberry Pi.

El siguiente paso, sería desarrollar el script que permita al módulo poder comunicarse con el Gateway, ya sea para recibir o enviar datos.

5. Líneas futuras

Como colofón a este proyecto, vamos a estudiar las posibilidades de la shield en un futuro próximo. Como ya hemos mencionado con anterioridad, este proyecto es solamente una parte de otro proyecto que lo engloba de mayor envergadura. Este proyecto busca llegar a tener un validador multiprotocolo que permita determinar, entre otras cosas, la calidad de la señal que los chips reciban. Una puesta en situación, por ejemplo, sería la de un operario que desea instalar ciertos sensores, como podrían ser de temperatura, humedad, etc. Para ello, debemos saber la calidad de señal que tenemos en un determinado lugar, para entre otras cosas, estudiar la viabilidad de la instalación de estos sensores, o en su defecto, si necesitamos aumentar la potencia de alguno módulo que actúe como nodo, o como ya hemos visto, como coordinador en el caso de que la calidad no sea lo suficientemente buena. Mediante este validador, el operario podría tener toda esta información, y de esta manera colocar estos sensores con certeza de que se podrán comunicar con el nodo o coordinador de forma satisfactoria.

Bien es cierto que para ello se necesitará el perfil de algún ingeniero informático que se dedique a la programación de los chips insertados en la placa, para que programe todas las funciones que el sistema requiera. Pese a ello, vamos a facilitar un poco la faena buscando algunas librerías que nos permitan trabajar con los chips instalados en la placa.

5.1 Lenguajes de programación

Para la programación de los módulos existen muchos tipos diferentes de lenguajes de programación, pero los más utilizados actualmente son Phyton o C/C++. Vamos a explica brevemente sus características principales, ya que posteriormente vamos a exponer algunas librerías basadas en estos lenguajes de programación.

En cualquier caso, ninguno es mejor que otro, sino que simplemente son 2 de los lenguajes más utilizados. El uso de uno u otro dependerá de la persona que deba realizar la tarea de programar los chips.

5.1.1 Python

Python es un lenguaje de programación muy versátil, multiplataforma, y que destaca por tener un código legible y limpio. Dispone de licencia de código abierto, lo que significa que no hay que pagar por su utilización. El objetivo principal es la automatización de procesos. Además dispone de una amplia biblioteca de recursos. Una de las ventajas principales del uso de Python es que está diseñado para expresar de forma muy clara las instrucciones que debe seguir un programa, sin necesidad de indicar detalles de bajo nivel, como podría ser, los tipos de variables, el tamaño de las estructuras de datos, o el manejo de la memoria. Evidentemente, no todo son ventajas, y estas características contrastan con que Phyton se ejecuta de forma más lenta que otros lenguajes de programación más tradicionales, que no requieren intérpretes, y que son ejecutados directamente por el procesador.

Es por ello, que para los próximos eventos, se debería tener muy en cuenta el uso de este lenguaje de programación.

5.1.2 C++

C++ es uno de los lenguajes precursores de la programación orienta a objetos. C++ hereda la sintaxis del lenguaje C. Se trata de uno de los lenguajes más utilizados, y por ello, se trata de un lenguaje muy potente en lo que se refiere a creación de sistemas complejos, por lo que se conforma como un lenguaje muy robusto. El hecho de programar en C++ obliga al usuario a conocer mejor el hardware del dispositivo, por lo que da una mayor flexibilidad y optimización.

5.2 Librerías

Actualmente, disponemos de muchos entornos de programación para el desarrollo de las funciones que nos ofrecen nuestros chips. Los lenguajes más utilizados para la programación de éstos son Phyton, C o C++. Además, cabe mencionar que la programación de éstos se puede realizar a través de la misma Raspberry, o en su defecto, también podemos emplear un arduino con su correspondiente shield, ambas opciones son viables.

Las principales librerías que se pueden utilizar para la programación de los chips, las vamos a mostrar a continuación.

5.2.1 Librería ArduPi

ArduPi es una librería C++ que permite desarrollar programas para Raspberry Pi como si estuviéramos desarrollando un programa con un arduino. Dispone de todas las funciones para controlar las comunicaciones del puerto serie, pines i2C, SPI y GPIO.

Esta librería podemos obtenerla desde el siguiente enlace:

www.cooking-hacks.com/skin/frontend/default/cooking/images/catalog/documentation/raspberry_arduino_shield/arduPi_1-5.tar.gz"

Una vez tenemos el archivo descargado en la Raspberry PI, debemos desempaquetarlo.

Una vez desempaquetado, nos encontramos con 3 ficheros: arduPi.cpp, arduPi.h y arduPi_template.cpp. Los dos primeros ficheros componen la librería propiamente dicha, mientras que el último fichero nos abre una plantilla que nos permite desarrollar la programación en C++.

También podemos utilizar la librería mencionada en el anterior apartado “Wiringpi” para la comunicación con los buses de la Raspberry Pi, aunque preferimos recomendar esta ya que es más completa, y además funciona tanto en Raspberry como en Arduino.

5.2.2 Librería Pynrf24

La librería Pynfr24 nos permite interactuar con el módulo NRF24l01. Se trata de una librería para Phyton, bastante actualiza y utilizada por muchos usuarios. Esta librería podemos obtenerla desde el siguiente enlace:

https://github.com/jpbarraca/pynrf24

Puesto que nuestro módulo NRF24l01 está conectado a través del puerto SPI, debemos añadir a esta librería una supletoria que nos permita comunicarnos con este puerto. Podemos obtener esta librería desde el siguiente enlace:

https://raw.github.com/doceme/py-spidev/master/setup.py

5.2.3 Librería para Xbee en Phyton

Si queremos interactuar con un módulo Xbee utilizando como lenguaje de programación Phyton, podemos utilizar una de las librerías más completas para Xbee. Esta librería podemos obtenerla desde el siguiente enlace:

https://github.com/niolabs/python-xbee

5.2.4 Librería LoRa Gateway

Para el módulo de LoRa tenemos una situación un poco diferente. Para una comunicación con LoRa, necesitamos configurar el Gateway, y posteriormente configurar cada nodo LoRa/LoRaWAN. Como ya hemos visto, las redes LoRaWAN se disponen en una topología de estrella, dónde los gateways retransmiten mensajes entre los nodos y un servidor de red. El Gateway se conecta al servidor mediante una conexión IP estándar, mientras que los nodos y el Gateway lo hacen mediante un enlace directo. Por lo tanto, necesitaremos una librería especial para la configuración del Gateway, que podemos obtener desde el siguiente enlace:

https://github.com/CongducPham/LowCostLoRaGw

6. Conclusión del proyecto

En la introducción del proyecto hablaba acerca de por qué había elegido este proyecto. Como así se comentó, me pareció muy interesante la idea de poder hacer algo “nuevo”, algo que es más propio de la rama de electrónica de telecomunicaciones.

Quizá no he sido la persona que ha ido a buscar aquello que ya conocía, y a partir de ello desarrollar un poco más. La verdad es que para mí ha sido un auténtico reto. Estoy acabando el grado este año, y estoy muy satisfecho de ver que tantos años de estudio y de esfuerzos valen la pena.

Desde que me presentaron el proyecto y comentamos cuales eran los objetivos de este, he dedicado prácticamente todos los días hasta entonces a intentar aprender, a intentar que todo estuviera listo, para finalmente poder entregar algo de lo que yo me sintiera orgulloso. Y la verdad es que lo estoy. Siempre pienso, que si hace unos meses me dicen que yo podría haber llegado a hacer esto con los conocimientos que tenía, y que lo haría funcionar, no lo hubiera creído, simplemente. Y han sido muchos los momentos en que he estado a punto de tirar la toalla, pensando que no hice una buena elección, y que por qué me metí en algo que apenas tenía conocimientos. Pero finalmente, con el debido apoyo, y con el debido trabajo las cosas acaban saliendo.

Me gustaría recalcar que, en todos estos años de estudio, lo que más se llega a desarrollar en uno mismo es la capacidad para “buscarse la vida”. La capacidad para no dejarse vencer, intentarlo una y otra vez hasta finalmente hallar el resultado deseado. Y puedo decir, muy seguro de ello, que así ha sido en mi caso.

Por un lado, pienso que lo aquí mencionado son las conclusiones a nivel personal que saco de este proyecto, sin duda, una actividad muy enriquecedora y que te pone en situación del día a día, y de los problemas con los que se encuentra un ingeniero, y que, como tal, debe procurar solucionarlos.

Por otro lado, y no menos importante, pienso que, una vez hecho este proyecto, se podría proponer hacer algo más de lo que se hace para el diseño de circuitos electrónicos. Mi experiencia me ha hecho ver, que a pesar de los conocimientos que tengo de electrónica, son muy insuficientes cuando uno quiere intentar realizar algo por su cuenta. Y es algo que he podido contrastar con más gente, incluso de la titulación de electrónica de telecomunicaciones. Por lo que también pienso que sería interesante plantear la posibilidad de hacer algunos cursos formativos para aquello que quieran aprender, que probablemente sean muchos. Y además se dispone de personal docente en la universidad que lo hace realmente bien.

Sin duda alguna, creo que ha sido un auténtico proyecto de ingeniería. Me ha llegado a poner al límite en algunas situaciones, y ha hecho sacar lo mejor de mí en todo momento para poder sacarlo adelante. Así que, muy orgulloso del trabajo realizado, y de la gran cantidad de conocimientos obtenidos.

7. Glosario

Shield Placas de circuitos modulares, que pueden montarse unas encima de otras, y que permiten a un dispositivo (en nuestro caso, una Raspberry Pi), dar funcionalidades extra

802.11 Conjunto de protocolos de comunicaciones inalámbricas.

802.11b Protocolo de comunicaciones inalámbricas.

802.11g Protocolo de comunicaciones inalámbricas.

IoT Internet de las cosas.

Gateway Dispositivo que actúa de interfaz de conexión entre dispositivos, puerta de enlace.

Escalable Hábil a la hora de reaccionar y adaptarse sin perder la calidad, maneja de manera fluida el crecimiento continuo de trabajo.

LPWA *Low Power Wide Area Networks*: área extensa y baja potencia.

UNB *Ultra Narrow Band*: tecnología que transmite sobre un canal de espectro muy estrecho, es decir, un canal con un ancho de banda inferior a 1KHz, para lograr un enlace de distancia ultra-larga (5 km en zona urbana o más de 25 km en zona suburbana.

ID Identificador del dispositivo.

3GPP Colaboración de grupos de asociaciones de telecomunicaciones, conocidos como miembros organizativos.

LTE *Long Term Evolution:* estándar para comunicaciones inalámbricas de transmisión de datos de alta velocidad para teléfonos móviles y terminales de datos.

2G Segunda generación de telefonía móvil.

Ericsson Compañía multinacional de Suecia dedicada a ofrecer equipos y soluciones de telecomunicaciones.

Sin licencia Bandas de radiofrecuencia que se pueden utilizar sin licencia, es decir, son libres y no se debe pagar.

PAN Red de área personal.

Nokia Empresa multinacional de comunicaciones y tecnología.

WiFi Mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.

IEEE Instituto de Ingeniería Eléctrica y Electrónica.

802.15.4 Estándar que define el nivel físico y el control de acceso al medio de rede inalámbricas de área personal con tasas bajas de transmisión de datos.

CSMA/CA Acceso múltiple por detección de portadora y prevención de colisiones.

M2M Conexión *"Machine to Machine".*

Latencia Suma de retardos temporales dentro de una red.

AEInnova Compañía al mando de este proyecto.

PCB Placa de circuito impreso.

Bootloader Gestor de arranque del dispositivo.

Nexperia Compañía que fabrica semiconductores.

Sparkfun Fabricante de electrónica integrada.

GitHub Plataforma de desarrollo colaborativa.

Pitch El pitch de un componente es la distancia que existe entre los diferentes pines o PADS de un componente.

PCBWay Fabricante chino de placas de circuito impreso (PCB)

XCTU Herramienta de configuración y prueba (XCTU). Software multiplataforma que permite interactuar con los módulos mediante un interfaz gráfico.

8. Referencias Bibliográficas

[1] IEEE Std 802.11-1997, *"Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications".*

[2] IEEE Std 802.11b-1999/Cor 1-2001, *"Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications speed Physical Layer (PHY) – Amendment 2: Higher extension in the 2.4 GHz band".*

[3] IEEE Std 802.11g-2003, *"Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications – Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band*

[4] J. Hallberg, M. Nilsson, K. Synnes, "*Bluetooth Positioning*".

[5] S.Ingram, "*UltraWideBand Indoor Positioning*".

[6] S. Tadakamadla, "*Indoor Local Positioning System for Zigbee, Based on RSSI*".

[7] *"Raspberry pi 40 pins"*https://www.raspberrypi-spy.co.uk/2014/07/raspberry-pi-b-gpio-header-details-and-pinout/

[9] "*Estándar UART*" https://geekytheory.com/puertos-y-buses-1-i2c-y-uart

[10] "*74LVC1G3157*" https://assets.nexperia.com/documents/data-sheet/74LVC1G3157.pdf

[11] "*Reglas y recomendaciones para el diseño de PCBs*" http://elektronikadonbosco.blogspot.com.es/2013/10/reglas-y-recomendaciones-para-el-diseno.html

[12] "*Convertidor de nivel lógico bidireccional"* https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookupguide?_ga=2.7034999.471563728.1514369101-1264174946.1513527226

[13] *"RN 2483"* http://es.farnell.com/microchip/rn2483-i-rm101/m-dulo-transcep-lora-433-868mhz/dp/2630124 //RN 2483

[14] "*XBP24B*" http://www.electronicoscaldas.com/modulos-rf/726-modulo-rf-xbee-serie-2-xb24cz7wit-004.html //

[15] *"NFR24L01"* https://www.luisllamas.es/comunicacion-inalambrica-a-2-4ghz-con-arduino-y-nrf24l01/

[16] *"Conceptos circuitos impresos"* http://www.pcb.electrosoft.cl/04-articulos-circuitos-impresos-desarrollo-sistemas/01-conceptos-circuitos-impresos/conceptos-circuitos-impresos-pcb.html

[17] *"NB-IoT"* https://www.esmartcity.es/comunicaciones/comunicacion-gestion-inteligente-residuos-urbanos-tecnologia-narrow-band-iot-nb-iot

[18] "*3GPP*" https://es.wikipedia.org/wiki/3GPP

[19] *"LTE"* https://es.wikipedia.org/wiki/LTE_(telecomunicaciones)

[20] "*2G*" https://es.wikipedia.org/wiki/Telefon%C3%ADa_m%C3%B3vil_//

[21] "*RN2483 datasheet*" http://www.alldatasheet.com/view.jsp?Searchword=Rn2483

[22] "*XBP24B datasheet*" http://www.alldatasheet.com/view.jsp?Searchword=XBP24B

[23] "*NFR24L01datasheet"*

https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf

[24] "*Reglas diseño PCB*" http://www.uni-kl.de/elektronik-lager/417703

[25] "*Reglas diseño PCB"* http://www.convertronic.net/Diseno/2013-10-31-09-58-46.html

[26] "*Tutorial KICAD*" http://docs.kicad-pcb.org/4.0.5/es/getting_started_in_kicad.pdf

[27] "*Tutorial KICAD II*" http://kicad-pcb.org/help/tutorials/

[28] "*Raspberry PI"* https://www.raspberrypi.org/

[29] *"WiFi-Ha-Low"* http://www.networkworld.es/movilidad/todo-lo-que-debes-saber-sobre-wifi-halow

[30] *"Bluetooth"* https://es.wikipedia.org/wiki/Bluetooth

[31] *"Nano-S100 datasheet"*

https://www.u-blox.com/sites/default/files/NANO-S100_DataSheet_%28UBX-16025707%29.pdf

[32] *"FiPy datasheet"* https://docs.pycom.io/chapter/datasheets/downloads/fipy-specsheet.pdf

[33] *"ATWILC 1000 datasheet"* https://www.microchip.com/wwwproducts/en/ATWILC1000

[34] *"RN1810 datasheet"* http://www.alldatasheet.com/view.jsp?Searchword=Rn1810

[35] *"XB8-DMUS-002 datasheet"* http://www.datasheetlib.com/datasheet/1286786/xb8-dmus-002_digi-international.html

[36] *"Gerber format"* https://en.wikipedia.org/wiki/Gerber_format

[37] *"pcbway Website"* https://www.pcbway.com/

[38] *"Phyton"* https://www.python.org/

[39] *"C++"* https://es.wikipedia.org/wiki/C%2B%2B

[40] "*Raspberry Pi Serial Connection*" https://elinux.org/RPi_Serial_Connection